CONSTRUCTING AND EVALUATING EXECUTABLE MODELS OF COLLECTIVE BEHAVIOR

A Dissertation
Presented to
The Academic Faculty

By

Brian Hrolenok

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Interactive Computing

Georgia Institute of Technology

December 2018

CONSTRUCTING AND EVALUATING EXECUTABLE MODELS OF COLLECTIVE BEHAVIOR

Approved by:

Dr. Tucker Balch, Advisor
School of Interactive Computing
Georgia Institute of Technology

Dr. Byron Boots
School of Interactive Computing
Georgia Institute of Technology

Dr. Magnus Egerstedt
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Greg Turk
School of Interactive Computing
Georgia Institute of Technology

Dr. Sean Luke
Department of Computer Science
George Mason University

Date Approved: October 15, 2018

To Charles Giering, my first teacher.

ACKNOWLEDGEMENTS

I would like to thank Professor Balch for his expert guidance, intuition, and most importantly for convincing me not to give up. Thanks to Professor Boots for believing I had an interesting problem, as well as helping me find the words (and math!) to convince everyone else. Thanks to Professor Egerstedt for asking the toughest questions, and challenging me to make bold claims and stand by them with conviction (and evidence!). Thanks to Professor Turk for thoughtful advice, rigorous scholarship, and for teaching me something new every day, which is why we do research in the first place. Thanks to Professor Luke for showing me how to write good research well, and for setting me on this path in the first place.

I have been very fortunate to be able to work with exceptional colleagues and friends whose support, advice, and conversations I would be at a loss without. I'd like to thank the current and former members of the Autonomous Robotics Lab at George Mason University, the Computational Perception Lab, and the Robot Learning Lab.

Finally, I would like to thank my family for their support and encouragement, without which none of this would have been possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Multiagent simulation (MAS) can be a valuable tool for biologists and ethologists studying collective animal behavior. However, constructing models for simulation is often a time-consuming manual task. Current state-of-the-art multitarget tracking algorithms can now provide high accuracy, high density tracking data of groups of research animals. Techniques from machine learning should be able to leverage the wealth of information such data provides in order to automatically find good models of animal behavior that can be executed in simulation. However, models trained using traditional single-step loss functions can lead to behaviors that are qualitatively dissimilar to the behavior of real animals, while Expectation Maximization (EM) methods computed over full trajectories are subject to local suboptima. These problems are compounded in the case of multiple interacting animals, particularly in highly cooperative species whose behaviors exhibit complex or emergent properties, generally termed *collective behavior*. This work describes methods for building executable models for these kinds of behaviors, the trade-offs between their strengths and weaknesses, and introduces a novel quantitative evaluation framework that addresses limitations of existing methods.

Collective behaviors can be broadly classified along (at least) two dimensions: stochasticity and statefulness, which are useful in illustrating the need for different techniques and evaluation criteria for different classes of behavior. Stateless behaviors are typified in the schooling behavior of fish, while foraging in ants is an example of behavior with state. In both behaviors, variance in the training data can be interpreted as either *noise* or *stochasticity*, and each interpretation admits different criteria for evaluation. Chapters 3 and 4 present specific techniques for building models of behavior with and without state following these examples, and also discuss the implications of the interpretation of variance, which naturally leads to a novel framework for evaluation of executable models of behavior that are inherently stochastic.

This framework combines measures of statistical similarity among a number of high-level characteristics of behavior, which is introduced in chapter 5 as the definition of *Behavioral Divergence*. Behavioral Divergence has a number of interesting and useful properties as a method for evaluating executable models, generally complementary to evaluation methods that focus on predictive accuracy such as single-step or trajectory error.

CHAPTER 1
INTRODUCTION AND BACKGROUND

This dissertation presents a framework for building and evaluating behavior models, specifically those models that are well suited to execution within some simulation engine or physically embodied agents, and classes of behavior that include complex interactions between agents. At the core of this framework is a method for evaluating such models, termed *Behavioral Divergence*, which serves as the central thesis of this document.

> ***Behavioral Divergence provides an effective means for evaluating learned executable models of collective behavior.***

While Behavioral Divergence is the common thread that connects all of the work presented here, there are several natural research questions that are addressed along the way.

1. How can executable models be automatically learned from data collected through observation?
2. What representations are sufficient for different classes of behavior?
3. How are other evaluation criteria related to Behavioral Divergence?
4. How can executable models be used once they are constructed?

So in addition to fully defining Behavioral Divergence and relating it to other evaluation criteria, subsequent chapters describe methods for learning behavior models from data, different classes of behavior and behavior models, and how executable models can be used to aid the development of data analysis tools.

This chapter is intended to both define terms, and to motivate the objective as a whole. The actual "how" of building (chapters 3, 4), evaluating (chapter 5), and using (chapters 2, 6) these models will be discussed in the remaining chapters. Following motivation,

basic definitions used throughout are presented. The next chapter will present a detailed motivating example where the social structure of a group of rhesus monkeys is inferred with the help of an executable model of behavior.

Motivation: modeling collective behavior

Biologists and ethologists have found a useful tool for describing and analyzing the behavior of social animals in agent-based models (ABMs) and multiagent systems. Such models have been successfully used to describe and analyze hunting behavior in ants (Yang et al. 2012), schooling behavior in fish (Couzin et al. 2002), and nest site selection behavior, a form of collective decision making, in both ants (Pratt et al. 2005) and bees (List, Elsholtz, and Seeley 2009). However, until now these models have been constructed by hand after careful analysis of large quantities of empirical observation data. For example, in constructing their model of collective nest choice behavior, Pratt et al. (2005) examine over 12000 interactions between approximately 290 ants in 12 videos each ranging from 30 to 150 minutes in length, by hand. The problem of automating the construction of these types of ABMs can be largely decomposed into two separate tasks: *tracking*, the process of converting observational data typically in the form of video recordings into accurate tracks of how the animal moved and interacted with its environment; and *constructing*, where the tracking data is used to create the executable model. Although multi-target tracking is an active area of research in the general case, the constraints on the tracking problem in this domain are sufficient that existing solutions such as those presented by Feldman, Hybinette, and T. Balch (2012) are sufficient, and so the main research focus in this dissertation is on the automatic construction and evaluation of ABMs from accurate tracking data.

Executable models

The goal of this work is to build and evaluate *executable models*, and so it is important to differentiate between these and the broader class of "models of behavior" used in many

disciplines. An executable model is a model that can be run step-by-step within a simulation engine to produce simulated actions. More formally, a model $f : \mathcal{S} \times \mathcal{O} \rightarrow \mathcal{A} \times \mathcal{S}$ is a mapping from internal state $s \in \mathcal{S}$ and external perception $o \in \mathcal{O}$ to immediate reactions $a \in \mathcal{A}$ and a (potentially) new internal state. These sets, $\mathcal{S}, \mathcal{O}, \mathcal{A}$, are specific to the particular agent, environment, and even class of behaviors of interest, but they generally have the following properties

- $\mathcal{S}$ is typically bounded and discrete. Although some natural parameterizations of internal state that are continuous or unbounded exist, realistic embodied agents are constrained by finite memory of limited accuracy.

- $\mathcal{O}$ mitigates the actual state of the environment to those things that are perceptible to the agent. For robots the capability of the agent is known in advance, but for animals no "ground truth" is available. Importantly, the perceived state of other agents is included here.

- $\mathcal{A}$ describes the immediate possible responses the agent can exhibit and is almost always fully prescribed by the granularity of the simulation being run and the mechanical limitations of the agent.

Simulating such a behavior in a given environment $\mathcal{E}$ requires a few more mappings in addition to f. The *sensor model*, $\phi : \mathcal{E} \rightarrow \mathcal{O}$ is a mapping from environmental states to agent perceptions. The *dynamics* of the environment $\psi : \mathcal{E} \times \mathcal{A} \rightarrow \mathcal{E}$ maps from the current configuration of the environment to a new configuration given an agent's action. For simulations with N agents, the dynamics should be a function of the actions of all agents since these could interfere with each other. With these preliminaries the process of simulation can be formally described:

1. Pick initial configuration for the environment and N agents: $e^{(0)} \in \mathcal{E}, \mathbf{s}^{(0)} \in \mathcal{S}^N$

2. Determine the reactions of each of the agents: $(a_i^{(t)}, s_i^{(t+1)}) \leftarrow f(s_i^{(t)}, \phi(e^{(t)}))$

3. Compute the new state of the environment: $e^{(t+1)} = \psi(e^{(t)}, a_1^{(t)}, \ldots, a_N^{(t)})$

4. Repeat from 2 until termination

This describes the general class of models of interest, but certain subsets of the general case illuminate some limitations with the naive approach to inferring a model given observational data, so it is useful to highlight them here:

- *Stateless models:* When f does not depend on any internal state, it is possible to write $f(s, \phi(e)) = f(\phi(e))$. This indicates that the behavior is a function of the agents current perception of the environment, and maintains no information of previous configurations, or history. While this is a very strong limitation, there are a number of behaviors that produce complex or emergent behavior with this property. For example, the "Boids" model due to Reynolds (1987) produces complex flocking behavior with no recourse to state. The converse case, when f depends strongly on s includes a much larger class of behaviors, of which the foraging behavior of ants is an example that is explored in detail in chapter 4. An important note: the term *state* is frequently overloaded in robotics and multiagent systems. Stateless in the context described here is specifically referring to what has been called *behavioral state* (Arkin 1998). The use of the term originates in finite state automata (FSA), when a version of the computational model was used as a representation method for composing multiple low-level behaviors into a hybrid high-level behavior (also called a behavioral assemblage). This document retains that sense of the word as the models discussed in 4 are composed in a similar way, but it should not be confused with other usages of state, such as environmental state. Specifically, whenever the term *stateful* or *stateless* is used in the remainder, it will be in reference to behavioral state: some notion of internal memory which modulates which of a number of possible lower-level behaviors is active at a given time. Other uses of "state" should be clear from context, and for the most part explicit (i.e. environmental state, behavioral

state, internal state, etc.).

- *Stochastic models:* When f does not describe a deterministic mapping from state and perception to action, but instead describes a distribution over possible actions, then the behavior it represents is *stochastic* or *nondeterministic*. It is intuitive to consider most behaviors as belonging to the converse case, when f is entirely deterministic, and this leads naturally to an obvious optimization problem which is solved by the naive approach to constructing these behavior models. However, there are compelling examples of inherently stochastic behaviors where this approach will fail. The difference between inherent stochasticity and measurement noise is an import one that will be discussed at length in chapter 3.

These two subsets can be thought of as orthogonal as an executable model may be both stateless and stochastic, neither, or any combination of the two.

Evaluating executable models

While chapter 5 discusses evaluation in more detail, it is worthwhile to pause here for a moment and set out some notation and common terminology. Learning problems are typically formalized in terms of minimizing some *loss*. In the case of executable models of behavior this loss should capture some notion of similarity between the actions performed by two models. Given a loss function $\ell : \mathcal{A} \times \mathcal{A} \to \mathbb{R}$ a suitable measure of how similar a learned model $\hat{f}$ is to a target model f would be the expected loss between them:

$$\mathbb{E}_{e \sim P}\left[\ell(\hat{f}(\phi(e)), f(\phi(e)))\right] \tag{1.1}$$

also known as the *risk* of $\hat{f}$ (with respect to ℓ and f). When learning, the intent is to choose $\hat{f}$ such that 1.1 is minimized. The key to equation 1.1 is in deciding what distribution P the expectation is taken with respect to. In standard supervised learning problems P is not known, but it is assumed that independent samples can be drawn from this distribution to

create training data. Crucially, any guarantees on the performance of the learned model are given in terms of the distribution of the training data.

However, in the formalism described in the previous section the actions taken at one timestep $a^{(t)}$ *directly affect* the state of the environment on the next timestep $e^{(t+1)}$, and it would be unrealistic to make an i.i.d. assumption about data collected from trajectories generated by such a system (Ross, G. Gordon, and D. Bagnell 2011; Ross and D. Bagnell 2010). In fact, it's not at all clear that the distribution of environmental states generated by the target behavior P_f would be equivalent to the distribution generated by the learned behavior $P_{\hat{f}}$.

Ross, G. Gordon, and D. Bagnell (2011) make the point that traditionally this change in distribution is ignored, and the learned model/predictor/policy is trained on samples from the target's distribution P_f, while the goal *should* be to use $P_{\hat{f}}$. The intuition that a learned model should be evaluated on states that it is likely to visit rather than states it might never see is compelling, but the root cause for the discrepancy is that $P_f \neq P_{\hat{f}}$. A natural alternative to equation 1.1 is to directly compare P_f and $P_{\hat{f}}$ using some notion of statistical divergence

$$D(P_f||P_{\hat{f}}) \tag{1.2}$$

where D is some measure of dissimilarity between two distributions, such as Kullback-Leibler divergence. The intuition behind this approach is that if two behaviors are similar, the distribution of states that they visit should likewise be similar.

In the remainder of this document, measures of error that are related to equation 1.1 will be collectively referred to as *predictive accuracy*, since they compare what actions the models would take in a given environmental state. Those based on some form of equation 1.2 will be referred to as *distributional similarity*.

Contributions

The remainder of this document will present support for the thesis statement given above, built up through a number of distinct contributions. These contributions are listed below, with citations for previously published work by the author if applicable:

1. Behavioral Divergence: an evaluation method for quantitatively assessing the similarity of observed behaviors based on the distributional similarity. (Chapter 5)

2. A method for learning executable models of stochastic behavior without internal state based on efficiently sampling from observations of a target behavior (Hrolenok and T. Balch 2013b; Hrolenok and T. Balch 2014). (Chapter 3)

3. A method for learning executable models with behavioral state by switching among low-level models (Hrolenok and T. Balch 2013a). (Chapter 4)

4. A method for inferring dominance and association preference structures in rhesus macaques from tracking data, developed using an executable model of behavior (Hrolenok, Maddali, et al. 2014). (Chapter 2)

5. Experimental results comparing the behavior of predictive performance based measures of error with Behavioral Divergence. (Chapter 6)

CHAPTER 2

THE IMPORTANCE OF AGENT BASED MODELS: A CASE STUDY IN PRIMATE SOCIAL STRUCTURE

In order to motivate the study of executable models of collective behavior, this chapter presents an example of how a hand developed executable model can be used to help design a novel data analysis method and tune its free parameters. In this instance, the model is developed entirely by hand, but the work flow, experimental design, and data collection are all carefully chosen to mimic the constraints an automated approach would have. Successive chapters will also follow this work flow, although the particulars of specific models will vary depending on the domain.

Agent-based modeling and simulation of animals solves two problems in the experimental study of animal behavior. First, the data collection cost associated with studies done in simulation using high fidelity models is relatively low, compared to the cost of running experiments and collecting data on real animal subjects. Using ABMs allows the researcher to run simulated experiments to increase the confidence of statistical analysis which might otherwise be less conclusive.

Second, when inferring model parameters directly from data, one is faced with the task of validation without access to any "ground truth". Performing the same inference methods on simulated data can provide crucial insight into how those techniques may perform on data from live animals. Both success and failure can be valuable clues into the capabilities and limitations of the inference methods.

This chapter focuses on the second of these; using ABMs as a validation tool. Section 2.3 introduces an agent-based model of social interactions between monkeys based on a well studied simulation with slight modifications relevant to the specific social measures. This ABM allows the development and quantitative assessment of the effectiveness of in-

ference methods designed specifically for recovering social structure from tracking data.

Historically, biologists and psychologists studying the social structures and dynamics of animals have relied on observation by trained researchers for the collection and coding of data, but developments in automated tracking systems have finally reached the accuracy required to recognize events of importance. It is now possible to record accurate, high-frequency spatial information over long periods of time. However, this qualitatively different kind of data requires a new approach to analysis.

As the sheer volume of data prohibits manual analysis, automated methods are necessary both for identifying key events and inferring relevant characteristics from identified events. This chapter examines how such automated methods can be applied in the specific case of inferring the social structure of a group of six rhesus macaque monkeys given tracking data of their movements over a period of three months at a rate of about 30Hz. In order to design and validate these automated methods, an agent based model of association and dominance behaviors is constructed to generate simulated tracks.

The remainder of the chapter is structured as follows. The next section reviews some related literature on social structure and agent-based modeling. The following section describes the specific aspects of social structure that these methods are intended to recover, and related behaviors. Next, the details of the approach to modeling and inferring social structure are presented. This chapter concludes with some results using simulated and real data, along with some high level analysis, conclusions, and directions for future work.

Motivation

Social structure in primate groups plays an important role in the health, behavior, and development of group members. Wallen (1996) has shown that social structure plays an important role in the development of behavioral sex differences, while Stephens and Wallen (2013) describe how social status can effect the actual physiological development of young monkeys. Sapolsky (2005) reviews how social status can effect a wide range of health

issues, both direct (such as access to resources), and indirect (stress related diseases). Being able to automatically infer the social structure of a group of animals then has wide ranging implications from maintaining the health and safety of laboratory animals, to determining the changes in social structure throughout the course of an experimental protocol.

Data is a crucial component in the process of developing automatic algorithms for inferring social structures. By taking an agent-based modeling approach, it is possible to create simulations of animal behavior that can generate synthetic "tracking" data with the appropriate characteristics, and so refine proposed algorithms while keeping data from live animals separate for validation. The work presented by Yang et al. (2012) has a similar goal, and provides a principled framework for using agent-based models to further the ethological study of foraging behaviors, specifically the foraging behavior of *Aphaenogaster cockerelli*. Hrolenok and T. Balch (2013a) presented work on learning these agent-based models of ant foraging directly from data using techniques from machine learning, and later (2013b) fish schooling, although there the purpose was the automated learning of the behavior model itself, while the focus of this chapter is in developing inference techniques using a known model. The development of this known model is heavily influenced by DOMWORLD, introduced by Hemelrijk (2000). Hemelrijk presents an agent-based model of dominance in primates that emerges spatial patterns typically found among certain types of rhesus macaque troops.

Social structures in rhesus macaques

One of the more intuitive measures of social structure in primates is association preference, which indicates which members of the group each individual prefers to spend time in close proximity to. A graph constructed from association preferences can illuminate subgroups, key individuals which connect otherwise disconnected groups (also known as cut vertices, or articulation points), as well as overall measures of group structure such as connectedness. The observable behavior where two or more monkeys spend time within relatively close

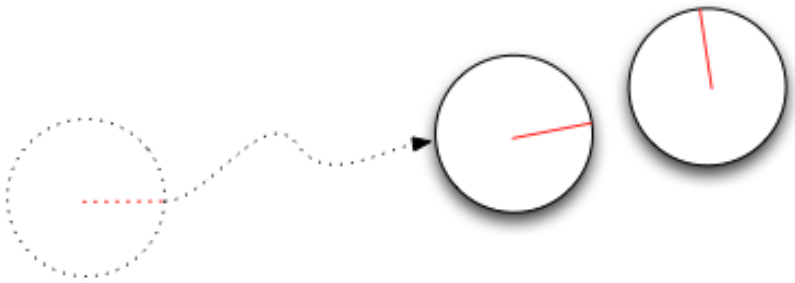

Figure 2.1: An approaching behavior that indicates an association preference. The strength of the indicated association preference is determined by the frequency and length of periods of close proximity.

proximity to one another indicates association preference. Figure 2.1 illustrates a grouping behavior that indicates association.

Another important measure of social structure is the dominance hierarchy. Dominance plays important roles both in interactions between individuals and group dynamics, and changes in dominance can indicate significant events of interest to the primate researcher. Observed displacement and withdrawing behaviors such as chasing and fleeing indicate a dominance relationship. Figure 2.2 illustrates a withdrawal behavior that indicates a dominance relationship.

While some association, displacement, or withdrawing behaviors rely on visual cues that the tracking system cannot directly sense (such as orientation), others can be detected directly from spatial data, as described in later sections. Spatial data of this kind was collected for 6 monkeys in a 3m x 3m enclosure over a period of three months by using a 3D position tracking system, the details of which are described by Huang et al. (2012).

Methodology

In order to be a useful tool for validation, the agent-based model was designed to incorporate the import behaviors mentioned previously, and parameterized so that recovered social

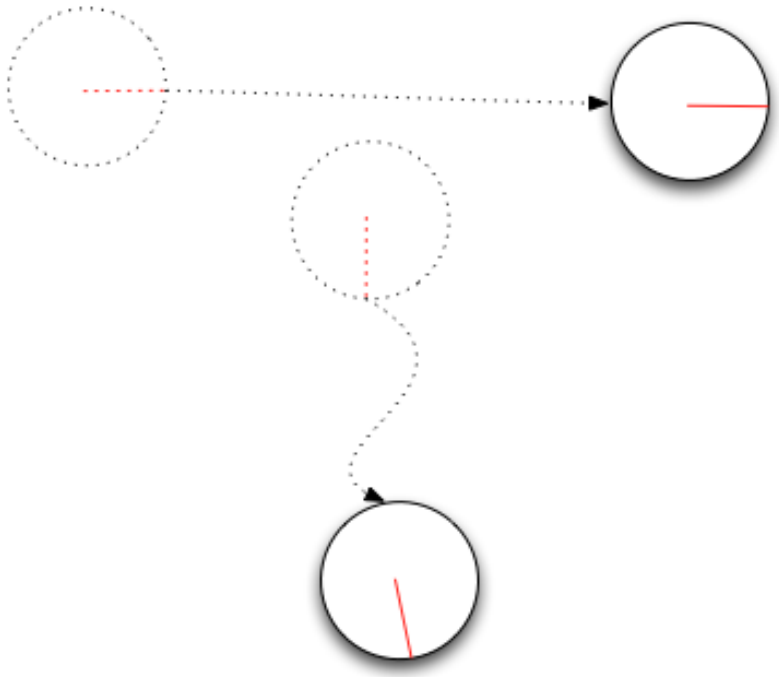

Figure 2.2: A withdrawal behavior that indicates a dominance relationship. The strength of the indicated dominance relationship is determined by the relative frequency with which each individual withdraws from the other.

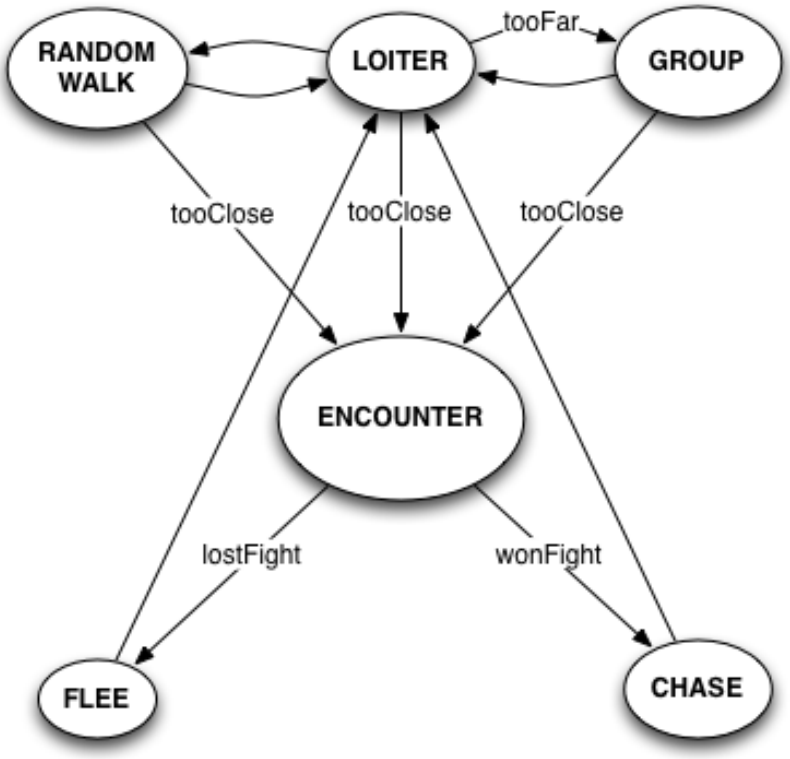

Figure 2.3: The SMALLDOMWORLD model.

structures could be compared with the "ground truth" of the simulation.

Agent based behavior model

The simulation model, called SMALLDOMWORLD is a modification of the earlier DOMWORLD of Hemelrijk (2000). The behavior of the individuals is guided by three components: a grouping component that draws individuals together, a dominance component where individuals confront each other and the winner chases the loser, and a random component where individuals wander about their environment at low speed.

In order to match the environmental conditions of the animals being studied, the DOMWORLD model presented in Hemelrijk 2000 was modified in a number of ways. In the troop being studied, the dominance relationships were already stable and established, where

as in DOMWORLD, dominance relationships are recalculated after every encounter. In DOMWORLD, as in the new model, dominance encounters only occur when an individual approaches another within some distance threshold representing an intrusion into personal space. In the new model, the probability of an intrusion on personal space resulting in a dominance encounter is given by the parameter σ where $\sigma = 1.0$ indicates a completely stable dominance structure with no confrontations, and $\sigma = 0.0$ ensures that any intrusion results in a confrontation. The new model used the same dominance confrontation mechanism as DOMWORLD: each individual is given a dominance weight, and the difference in weights probabilistically determines the winner of any encounter (see Hemelrijk 2000 for details).

As DOMWORLD does not attempt to describe association preference, the new model also introduced some selectivity into the grouping behavior. Grouping in DOMWORLD represented a desire by all individuals in the group to remain within some proximity of other group members, and so when an individual found itself far away from the center of the group, it selected another visible member of the group uniformly randomly to head towards. In the new model, each individual has a list of association preferences $\pi = \langle \pi_1, \pi_2, \ldots, \pi_n \rangle$ which can be thought of as the distribution over individuals selected for grouping. This leads to patterns of association which are non-uniform and give rise to the social structures described earlier. The list of association preferences can be combined into a single association preference matrix P with each row corresponding to a single individual's association preferences.

While these two modifications represent the most important changes between SMALLDOMWORLD and DOMWORLD, there were also several changes made to accommodate differences in the modeled environment and simulation engine. DOMWORLD's environment is long-range, discrete, and unbounded (toroidal in implementation), whereas the environment in this study is quite small, continuous, and interactions with the boundary of the enclosure are common (which necessitated some kind of obstacle avoidance behav-

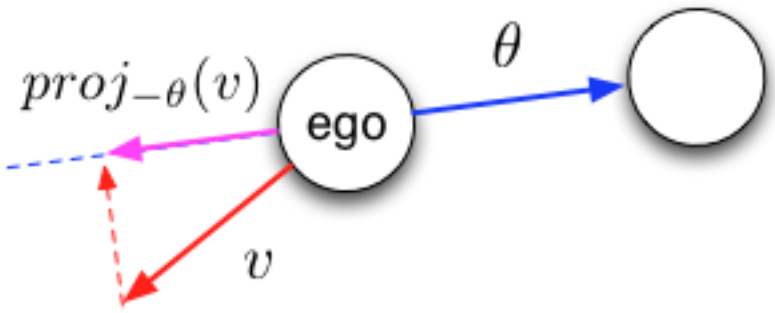

Figure 2.4: Detection of fleeing events.

iors). Figure 2.3 gives a graphical representation of the behavior model of individuals in SMALLDOMWORLD.

Heuristic behavior recognition

This section presents two heuristic methods for identifying association and dominance behaviors, and how they can be used to infer the social structures of a group of monkeys.

Time spent within proximity is a straightforward way to detect behavior which indicates association preference among group members. By counting the frequency and length of events where the *ego* — the individual whose preferences are being determined — comes within a threshold distance of another individual, and remains there at low to zero velocity for at least some minimum period of time, the individuals the ego prefers to spend time with can be inferred. If E_{ij} denotes the time monkeys i and j spend near each other, then the entries of the association preference matrix P can be computed as:

$$P_{ij} = \frac{E_{ij}}{\sum_k E_{ik}} \tag{2.1}$$

While detecting all types of withdrawal events may be difficult, a certain subset can

be captured fairly easily. One type of withdrawal involves the ego rapid moving directly away from the target, which are denoted *fleeing* events. Fleeing events can be detected by counting the length and number of events where the magnitude of the ego's velocity (v) projected onto the bearing (θ) between the ego and target is larger than a threshold (f), which is shown graphically in Figure 2.4. For each pair of individuals a dominance measure d can be computed as

$$d_{AB} = \left|\{e_{AB} \mid proj_{-\theta(A,B)}(v_B) > f\}\right| \tag{2.2}$$

If individual A flees from individual B more frequently than the opposite ($d_{BA} > d_{AB}$), then it can be inferred that A is subordinate to B. Choosing the threshold f effectively sets the sensitivity of this metric. By choosing conservatively, only a subset of fleeing events are used in the inference. In practice, picking f to correspond to roughly 30 degrees on either side of moving directly away from the target worked well.

Experiments in inferring social structure

Four experiments were performed to test the approach described above, three using data collected from the simulation model SMALLDOMWORLD with different parameterizations of the social structure, and one using the three months of tracking data collected from a small group of animals. The purpose in performing the simulation experiments was to measure the accuracy and robustness of the recovered social structure, and to characterize scenarios where the method might not be able to recover social structure. Each of the simulation experiments was replicated ten times using the same parameter settings with different initial configurations and random seeds.

The first experiment simulated a single group of monkeys with a social structure split into two disconnected subgroups, or cliques, as shown in Figure 2.5 and Figure 2.6. The parameterization which realized this structure had each individual's association preference

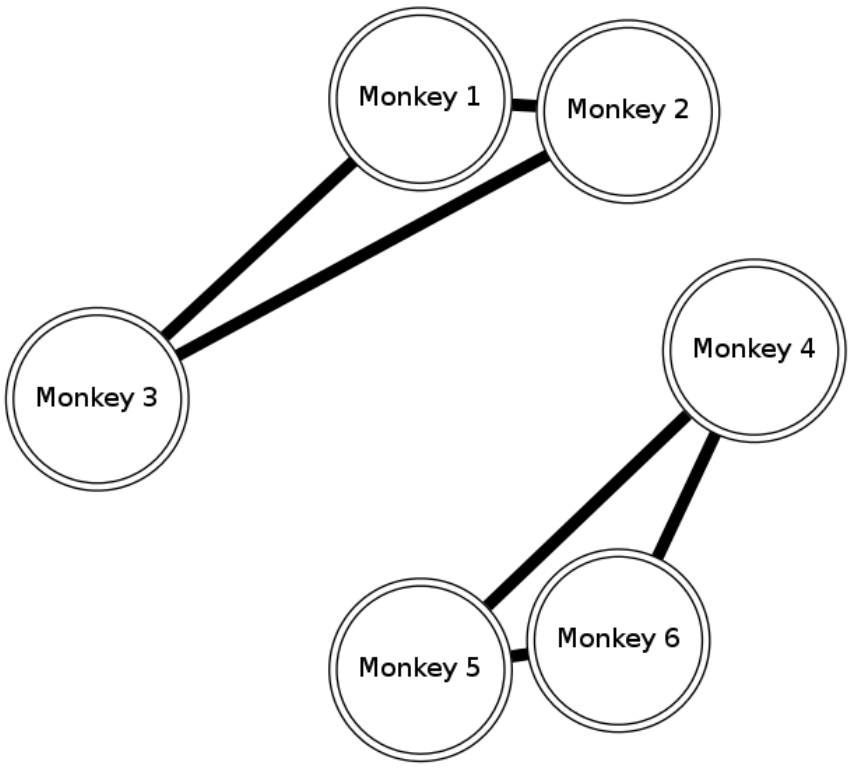

Figure 2.5: Association preferences for the disconnected scenario. This shows the ground truth graph. Compare with Figure 2.6.

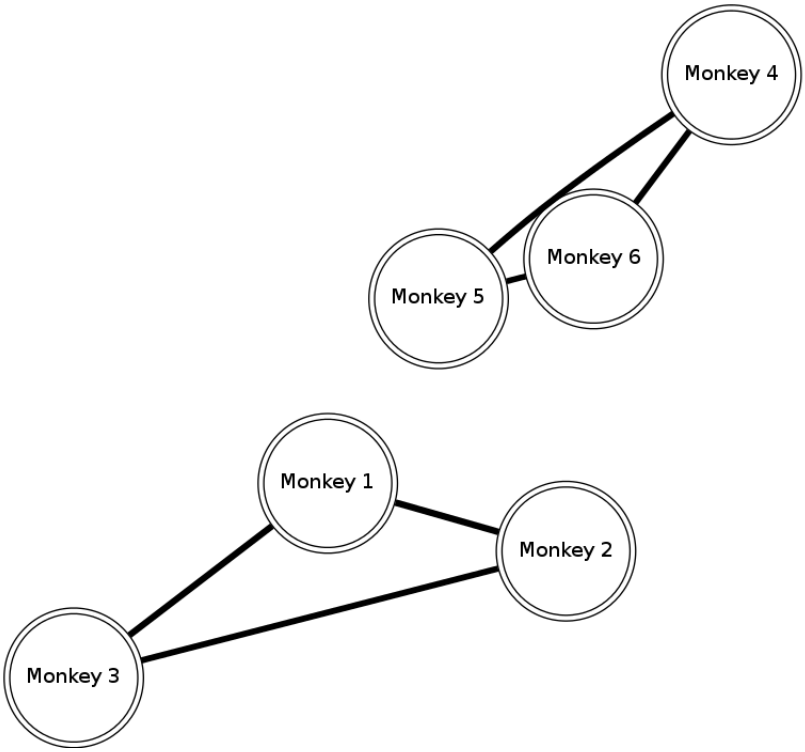

Figure 2.6: Association preferences for the disconnected scenario. This shows the recovered graph, which closely matches the ground truth, indicating that the association preferences are being correctly recovered. Line thickness corresponds to strength of association preference. Association preferences that fall below the threshold τ from Equation 2.3 are not shown.

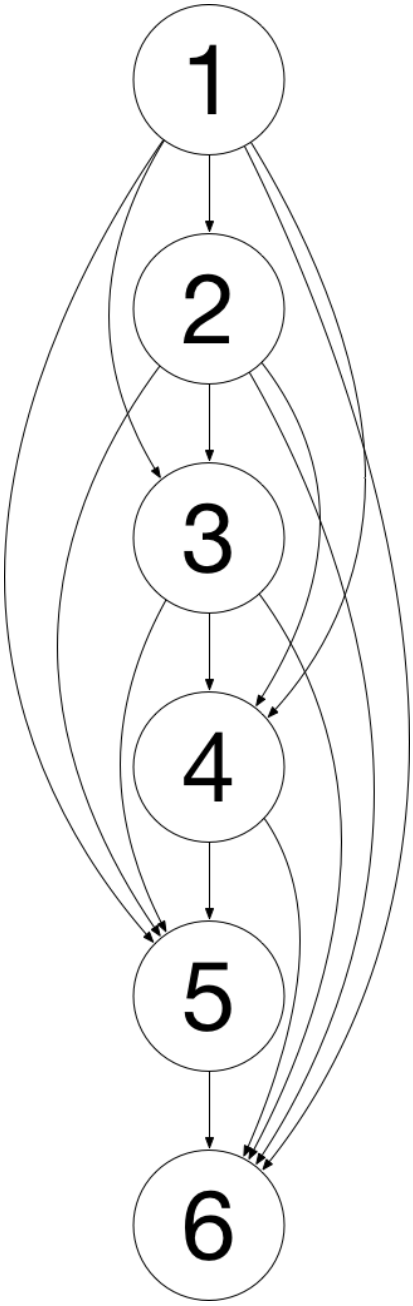

Figure 2.7: Dominance hierarchy.

Table 2.1: Simulation parameters common for all experiments.

Personal distance	0.25m
Near distance	0.8m
Fleeing speed	2.0m/s
Chasing speed	1.0m/s
Grouping speed	0.25m/s
Wander speed	0.12m/s

Table 2.2: Frobenius error of recovered association preference as compared to a randomly generated symmetric, normalized matrix with zero diagonal. This indicates that the recovered preferences are significantly more accurate than would be expected by chance. Reported values averaged over 10 runs.

Recovered AP	Avg. error (std.)	random AP
disconnected	0.1744 (0.0014)	0.2408 (0.0326)
neutral hinge	0.1002 (0.0015)	0.1797 (0.0350)
preferred hinge	0.1388 (0.0004)	0.1869 (0.0158)

set to 1.0 for other members of its subgroup, and 0.0 otherwise. This way there should be no deliberate preference to spend time in proximity of non-subgroup members. The dominance relationships for this and the two following simulation experiments was a direct linear relationship with rank corresponding to ID, as illustrated in Figure 2.7. The parameterization that realized this model set the dominance weight for the least dominant individual to 1.0, with each individual higher in the chain having twice the dominance weight as the next lowest, or ($D = 2^{N-i}$). The same dominance weights were used in each experiment. Hierarchy stability was set fairly high ($\sigma = 0.8$), so that dominance interactions were not frequent, but still frequent enough to reliably detect the dominance hierarchy. Other simulation parameters are listed in Table 2.1 and were estimated from tracking data of live monkeys where appropriate, or taken from the literature when available.

With the stated parameters, it was possible to consistently recover both the dominance relationships and association preferences in this experiment. One appropriate measure of the accuracy of the recovered association preferences is to take the Frobenius norm $\|\cdot\|_{\mathrm{F}}$ of

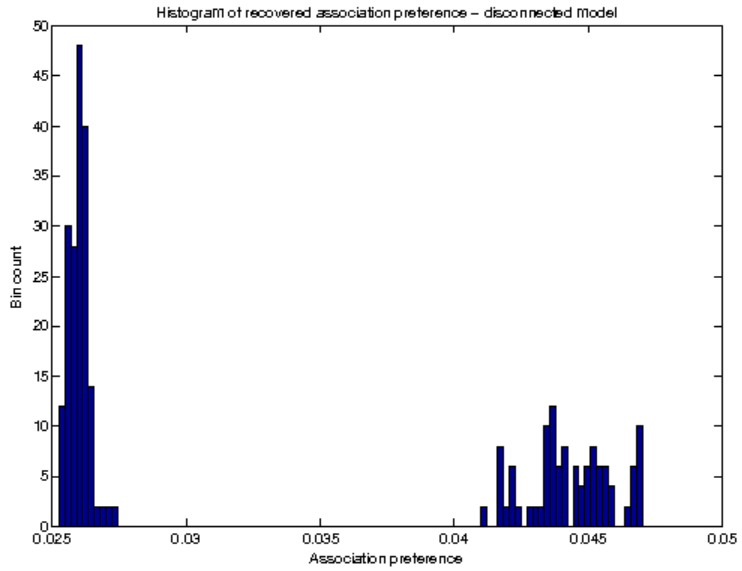

Figure 2.8: Histogram of association preference values recovered from the disconnected scenario. Notice the clear separation into two modes of the recovered association preferences.

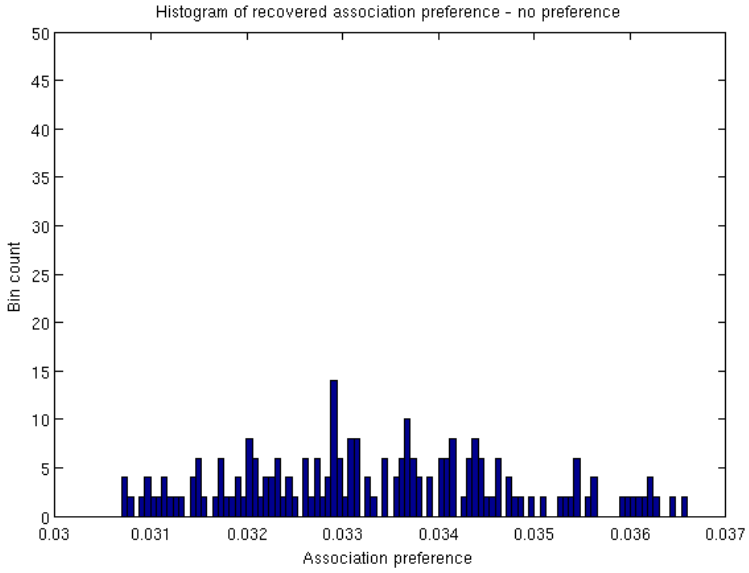

Figure 2.9: Histogram of association preference values recovered from a simulation with no association preferences. In this simulation, agents followed the same behavior model as 2.8, except when choosing to group they chose among neighbors without preference. This distribution has no distinct separation between clear modes.

the difference between the recovered preferences (P) and the ground truth (P^{GT})

$$\left\|P - P^{\mathrm{GT}}\right\|_{\mathrm{F}} = \sqrt{\sum_{i,j} \left(P_{ij} - P_{ij}^{\mathrm{GT}}\right)^2}$$

By comparing the norm of the difference from the ground truth for recovered preferences or a randomly generated matrix restricted to the same form (row-normalized, zero-diagonal, symmetric), it is possible to get a sense of the quantitative accuracy of the recovered preferences. The results of this comparison are given in the first row of Table 2.2, which shows that the recovered parameters are significantly closer to the ground truth than random.

In order to recover the graph structure shown in Figure 2.5, a threshold τ is chosen such that edges larger than τ are included in the graph while those smaller are not. The distribution of values in the association preference matrix, shown in Figure 2.8, gives a sense of whether τ can be chosen reliably. The distribution is clearly divided into two modes, as compared to the distribution of association preferences recovered from a simulation without any association preference shown in Figure 2.9. This indicates that by picking a a threshold between the two modes, the recovered graph will be stable to noise in the estimation of association preferences. In testing, picking

$$\tau = \frac{1}{n^2} \sum_{i,j} P_{ij} \tag{2.3}$$

where n is the number of agents, worked reliably.

In the second experiment the association preferences were modified so that one individual, denoted as the *hinge*, became an articulation point linking the two subgroups, as shown in Figure 2.10. To do this, the association parameters were modified such that the hinge individual preferred everyone equally, but no one had a preference for them. In terms of the parameterization, the hinge individual's row was set to $P_{hj} = 1.0, \forall j$, and its column $P_{ih} = 0.0, \forall i$. Results are shown in the second row of Figure 2.2. Again, as shown in

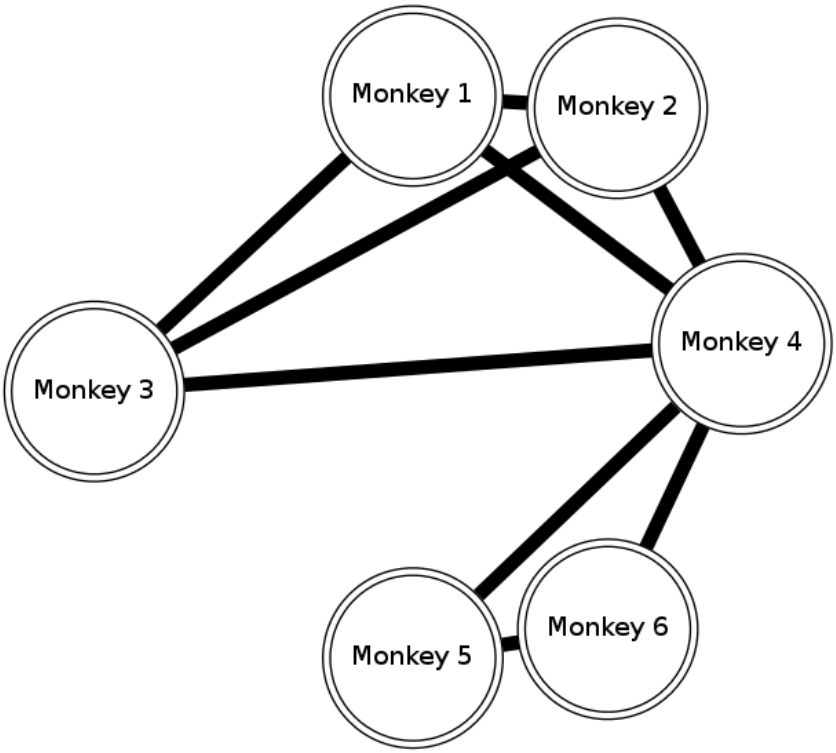

Figure 2.10: Ground truth association preferences with hinge node as used in the simulation.

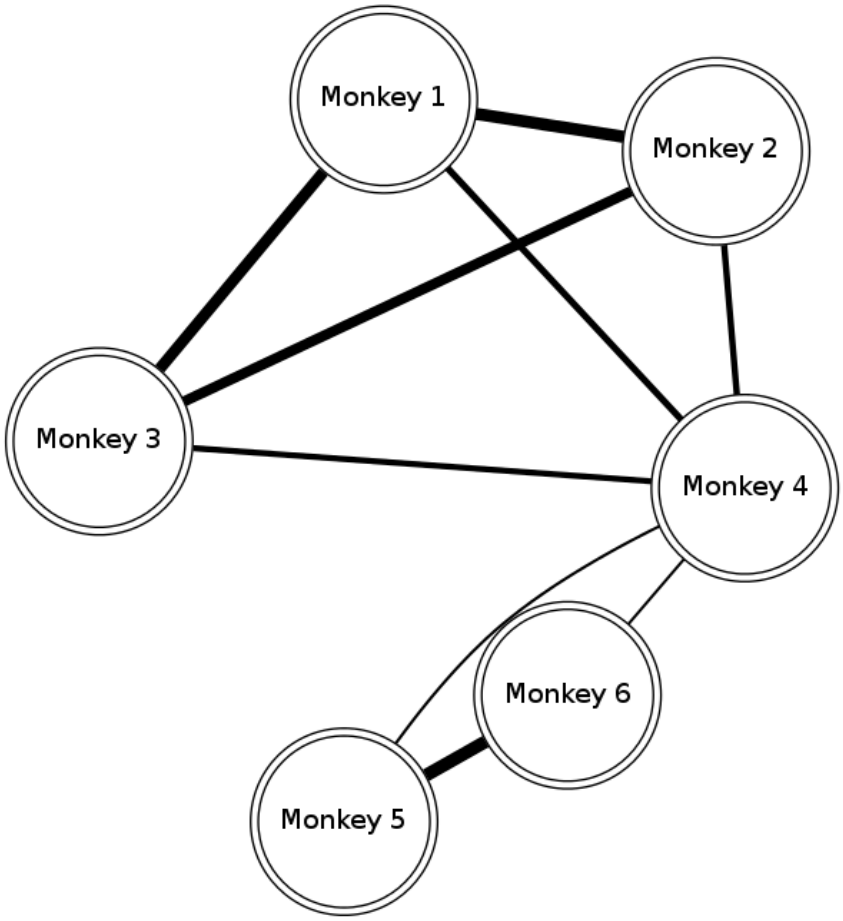

Figure 2.11: Recovered association preference, which closely matches the ground truth association preferences used in the simulation as shown in 2.10. Line thickness corresponds to strength of association preference. Association preferences that fall below the threshold τ are not shown.

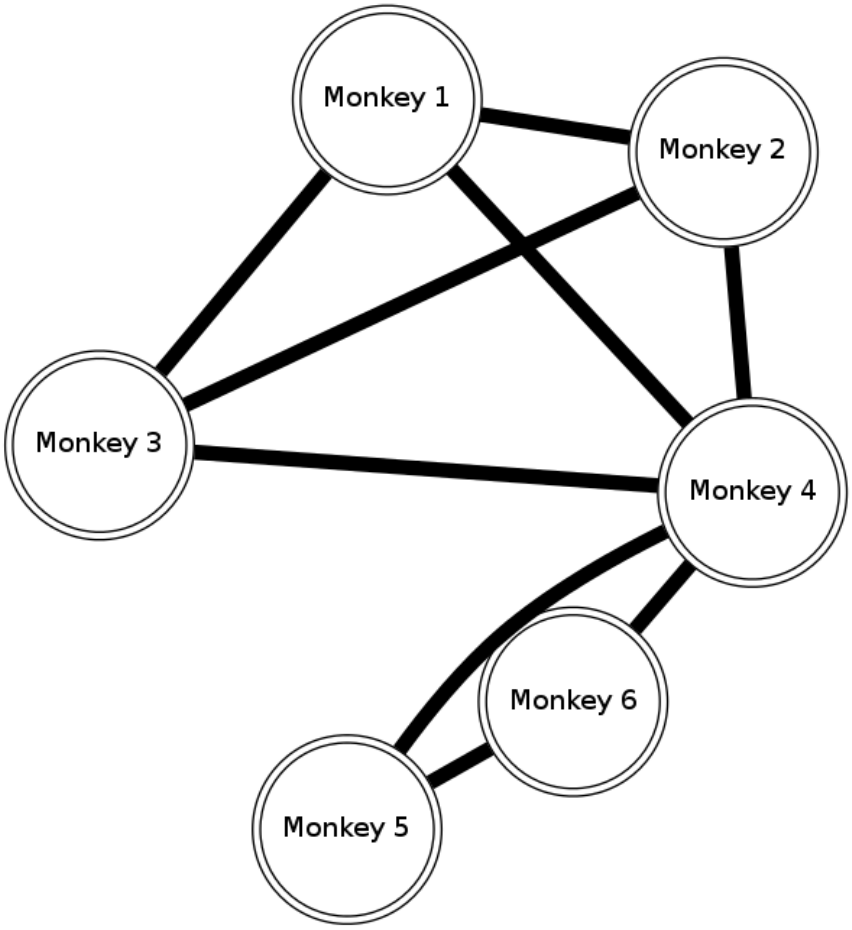

Figure 2.12: Association preferences for the hinge node scenario using non-transitive preferences. Monkey's 5 and 6 have a strong preference for Monkey 4, but not 1, 2, or 3. The non-transitive nature of the association preference means that there is a high likelihood for individuals with no association preference for one another to be within close proximity if they choose to visit the hinge node at the same time.

Figure 2.11 the recovered association preference is significantly closer to the ground truth than a random association preference, and the dominance hierarchy was recovered without error.

The third experiment repeated the previous experiment, but also allowed the other individuals to preferentially group with the hinge individual by setting $P_{ih} = 1.0, \forall i$. Doing this highlights a potential scenario where this method may not be able to recover the social structure accurately, specifically non-transitive association preferences. Note that the metric for association preference makes no distinction between individuals which are within

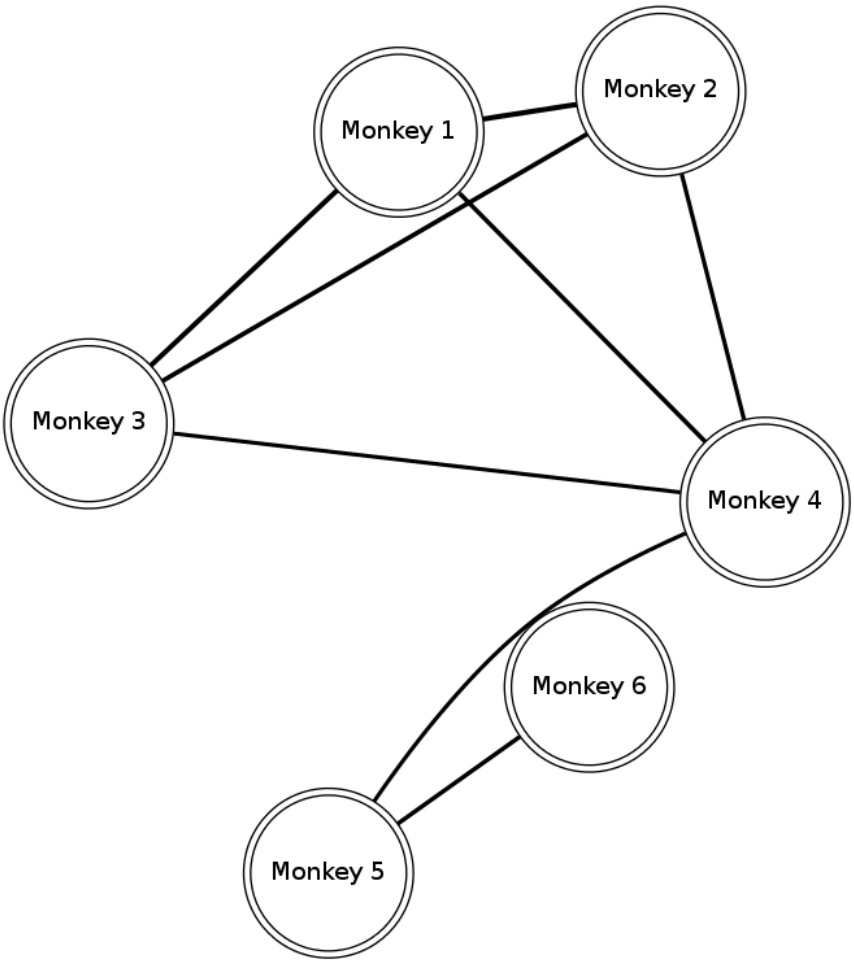

Figure 2.13: Recovered association preferences. The recovered graph is missing a link between monkey 6 and monkey 4 in the actual association preference graph (2.12). Notice also that the magnitude of the preferences — shown by the thickness of the edges — is much closer to the threshold value τ. Picking smaller τ results in additional edges that are not present in the simulated behavior. The non-transitive preferences make it difficult to choose a stable τ.

proximity because they chose to be, and those that just happen to be nearby. For example, if individuals A and B do not have any preference for association, but each has a high preference for associating with a third party C, then *regardless* of C's preferences, A and B will spend a high proportion of time in proximity of one another. Figure 2.12 and Figure 2.13 as well as the third row of Figure 2.2 illustrate how recovery performance is degraded in this type of non-transitive scenario.

Finally, for the fourth experiment the association preference recovery method and dominance structure recovery method were applied to tracking data taken of live animals. These tracks were collected with a tracking system based on time-of-flight multilateration of tags embedded in collars worn by the monkeys. Details of the tracking system are given in (Huang et al. 2012). Figures 2.14 and 2.15 show the recovered association preferences and dominance hierarchy for the entire period the monkeys were tracked.

The threshold parameter τ was chosen using the same approach described above, although from examining the distribution of association preferences shown in Figure 2.16, it is clear that this choice will be less stable with respect to noise. That is, it is more likely that some edges will be included or excluded from the graph due to small changes in association preference. The dominance relationship is a linear hierarchy $(4, 3, 5, 2, 6, 1)$ with individual 4 being the most dominant, and individual 1 being the least dominant. This agrees with the recovered association preference, where individuals are shown as preferring to associate with other individuals at similar ranks in the hierarchy.

Distributional characteristics and stochastic behavior

This chapter has demonstrated how a hand-built executable model of behavior such as SMALLDOMWORLD can be used to assist in the development of data analysis techniques. In the process, two factors which will be key in the remainder of this dissertation have been highlighted. First, the distribution of association preference was *characteristic* of the behavior in question. Figures 2.8 and 2.9 are markedly different, even though they

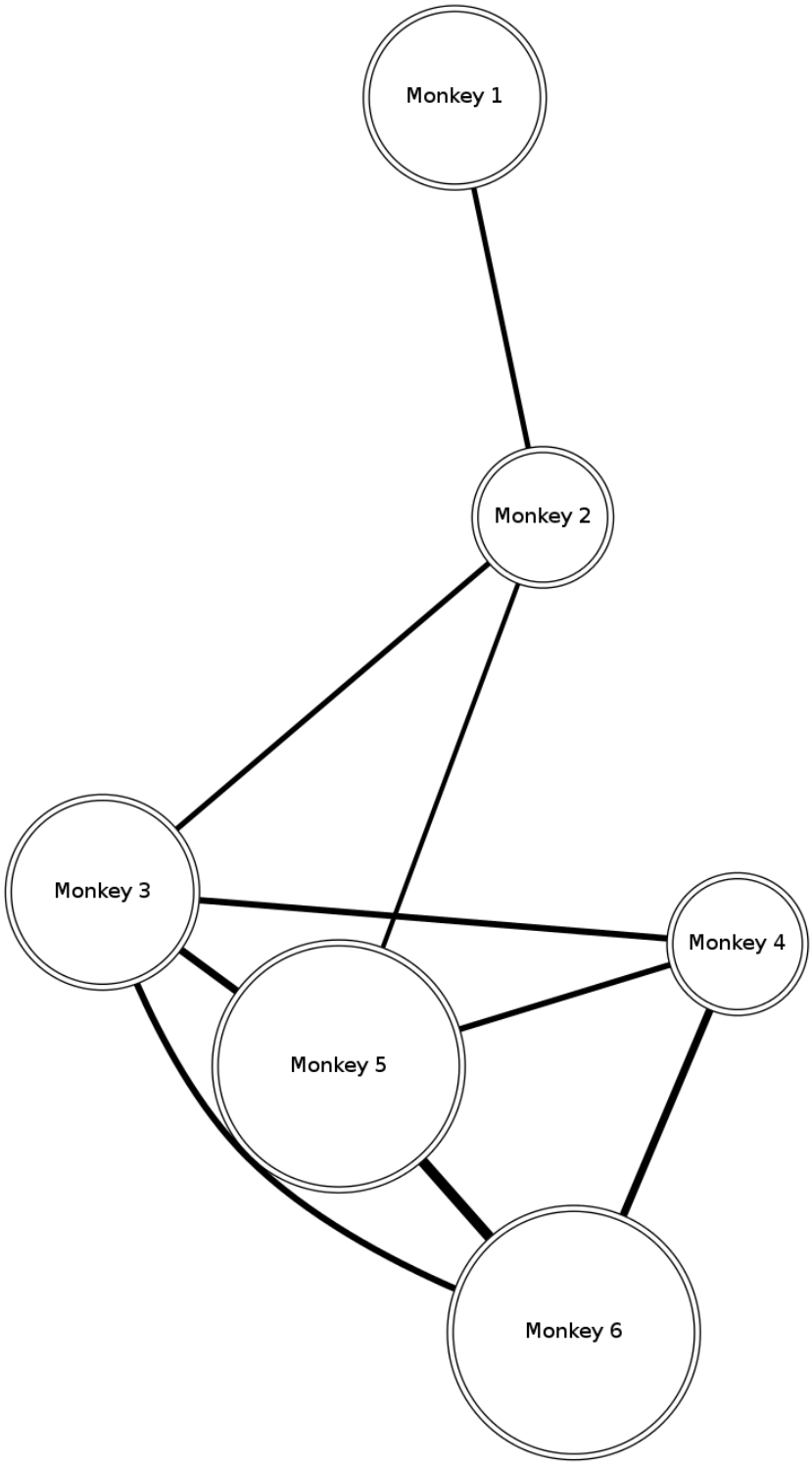

Figure 2.14: Association preferences for live animals. Diameter of the node is determined by the sum of association preferences for that node. Links are included if they are larger than the mean association preference, and their width is determined by how strong the association preference between the two nodes is.

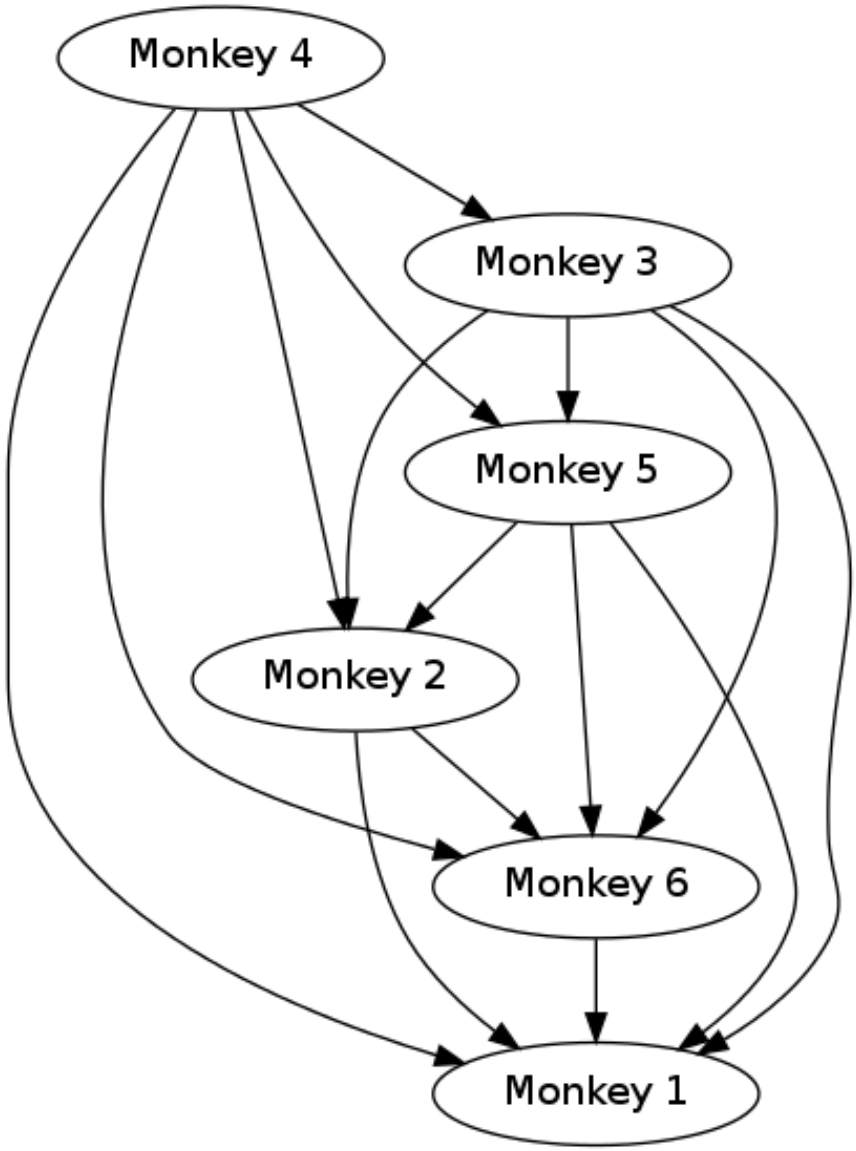

Figure 2.15: Dominance hierarchy for live animals. Linear chain hierarchies match with our simulated model of dominance behavior, but it is important to note that no part of our inference of dominance relationships *enforces* linear chains. So if the live animals had been an egalitarian troop with little to no aggressive displays, or if the dominance rankings had not been established, we would expect to see a different kind of dominance structure.

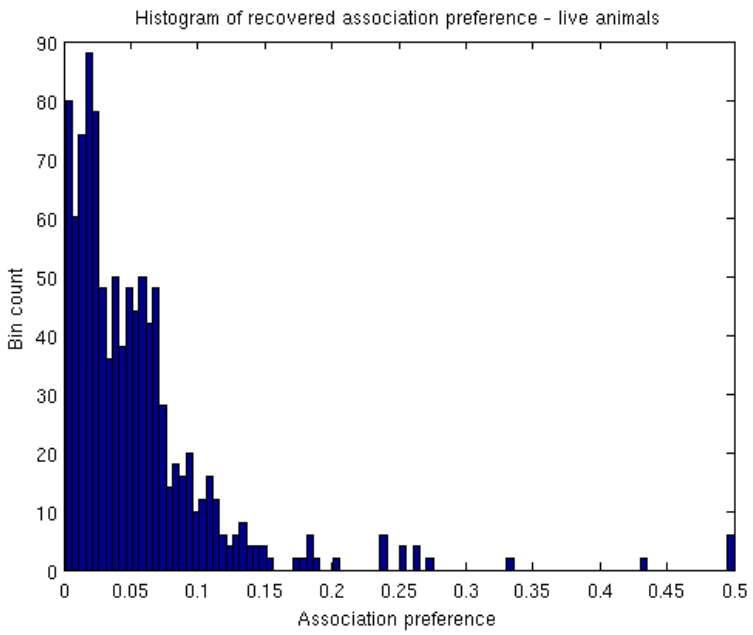

Figure 2.16: Histogram of association preference values recovered from the live animals. The secondary mode in this distribution is less distinct, but there still is a separation into low and high preference levels.

were generated by the same behavior model with slight differences in parameterization. This suggests that an appropriate quantification of the similarity or difference between characteristics of behavior can be used to *evaluate* how similar two behaviors are, a point that will be returned to in subsequent chapters.

Second, the models discussed in this chapter are all *inherently stochastic*, that is they represent behaviors in which variability in the response to a given set of perceptions does not indicate noise, but some fundamentally non-deterministic process. In DOMWORLD and SMALLDOMWORLD, whether a dominance interaction occurs, the result of any such interaction, and the choice of neighbor to group towards are all determined probabilistically. If inherently stochastic models are useful for modeling collective behavior, it is natural to ask what methods are appropriate for *learning* such models. Specifically, whether methods designed to eliminate or reduce variability are appropriate, and if not what alternatives exist. These two questions are the focus of the next chapter.

CHAPTER 3

LEARNING STOCHASTIC EXECUTABLE MODELS

The focus of this chapter is on the process of learning executable models of collective behavior from tracking data, specifically models that are non-deterministic and do not have internal state. While the class of behaviors that can be represented by models without internal state is more restrictive, it includes some important and interesting specific examples such as flocking or schooling behavior. Additionally, the mechanisms developed for these models can be used as a basis for more complicated models which do incorporate state, which is explored in detail in chapter 4.

Deterministic and stochastic behaviors

Consider the behavior of an agent which reacts to its environment in a deterministic way. It is reasonable to cast the problem of predicting the behavior of the agent as a supervised learning problem, where the function to be predicted $f : \mathbb{R}^d \to \mathcal{A}$ is the behavior of the agent, which is a function of some (known) features $\phi : \mathcal{E} \to \mathbb{R}^d$ of the current state of the environment $e \in \mathcal{E}$ that are relevant to the agent's behavior. In this approach a predictor $\hat{f}$ of the true behavior f is learned using a set of training data

$$D = \{(\phi(e_i), a_i) : a_i = f(\phi(e_i))\} \tag{3.1}$$

where each a_i in the dataset is the observed action of an agent, in response to perceiving $\phi(e_i)$. Such a dataset can be collected by observing the behavior of the agent and the state of the environment. The key design decisions in this formulation are the features ϕ, the form of the model $\hat{f}$, and the learning procedure that fits $\hat{f}$ to f given D.

The selection of ϕ, otherwise known as *feature selection*, is an important topic in and of

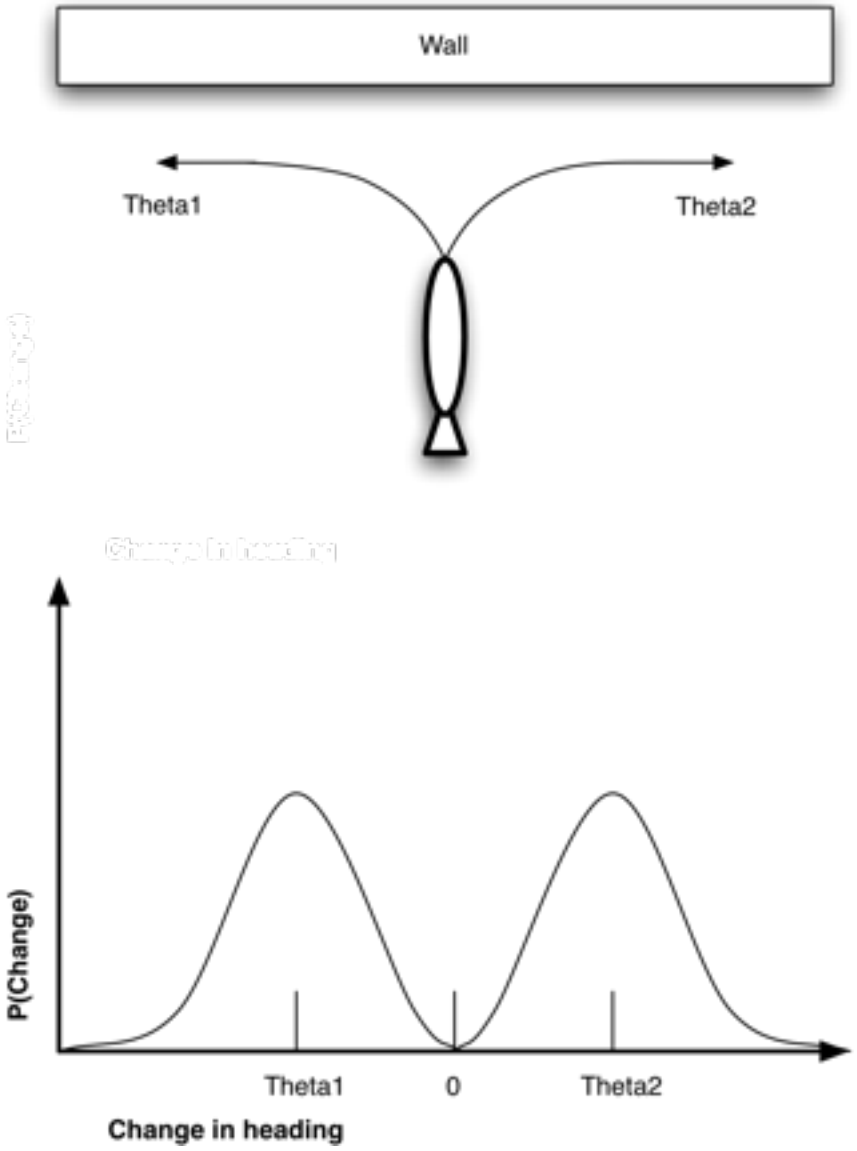

Figure 3.1: Graphical representation of a realistic distribution of behavior who's expected value has low probability. The observed frequency of turning left versus right are equal, so the distribution of behavior has two equal modes. The mean θ, which minimizes expected error, has very low probability, and if used in an executable model would generate unrealistic behavior.

itself, as it has a significant impact on the maximum possible performance of $\hat{f}$. However, good feature selection typically depends on domain knowledge specific to the problem at hand. For the remainder of this chapter, ϕ will be considered as given.

If the observations are noise free, enough data can be collected, and the predictor $\hat{f}$ has the expressive capability to reproduce f, then it is possible to simulate what the agent would do for any given environmental state e, and this simulation should reproduce the observed behavior as well as accurately predict behavior in unobserved states. Of these three assumptions, the least reasonable is that of noise, which is addressed next.

In the typical supervised learning setting, training data D is assumed to be noisy in the sense that some uncontrollable aspect of the data collection process introduces an unknown (but small and normally distributed) amount of error to the true output of f, that is,

$$D = \{(\phi(e_i), a_i) : a_i = f(\phi(e_i)) + \varepsilon_i\} \tag{3.2}$$

where $\varepsilon_i \sim \mathcal{N}(0, I\sigma)$. Frequently, it is assumed that variance in the observed behavior for a given point in the feature space is due to the noise process, and as a result many learning algorithms use estimates of central tendency which are robust to noise. The least-squares solution for linear regression is one such example, where the predicted output for an unobserved state e would be

$$\hat{f}(\phi(e)) = \langle \hat{W}, \phi(e) \rangle \tag{3.3}$$

where $\hat{W}$ are the weights found by the least-squares solution. This estimate minimizes expected difference between the predicted output and the training data, which is achieved by predicting along a line as close to the central tendency of the training data as is possible, under the given assumptions that f is linear, and the noise process is normally distributed, and importantly that the underlying behavior is in fact deterministic.

If the behavior is instead *stochastic*, that is, it contains some unpredictable element that

can be treated as non-deterministic, the variance in the observed behavior is not completely due to the noise process and should be accounted for by $\hat{f}$. In this case it is more natural to think of f as a distribution over the space of possible actions $\mathcal{A}$. The observations of the agent's behavior are then seen as samples drawn from f:

$$D = \{(\phi(e_i), a_i) : a_i \sim f(\phi(e_i))\} \tag{3.4}$$

Note that it is possible to cast the "noisy observation" formulation in (3.2) to this stochastic behavior version by assuming that f takes a particular form. For example, in the case of least squares linear regression, $f(\phi(e)) = \mathcal{N}(\langle W, \phi(e)\rangle, I\sigma)$, while keeping $\hat{f}$ as defined in (3.3). This highlights an issue with using fixed estimates as predictors for behavior. If f is symmetric, unimodal, and falls to zero rapidly as distance from the mode increases, having $\hat{f}$ predict a behavior near the mode (as in (3.3)) will often be a good estimate, and so can be fixed for any specific $\phi(e)$. However, if f is multimodal, asymmetric, or has "fat tails", the probability of behaviors far from any central tendency of f can grow significantly, and while the sample mean may in fact minimize the predictive error, the probability of behaviors near the mean may be arbitrarily small.

To illustrate this, take for example the behavior of a fish swimming in a shallow tank of water, which is illustrated in Figure 3.1. As the fish approaches the wall of the tank, it can choose to turn left or right. From the perspective of the fish, it does not matter which direction it turns, so long as it does not collide with the wall. The argument could be made that a deterministic behavior is still feasible since the probability of approaching a wall exactly along the perpendicular is effectively zero, but it is also true that fish can only perceive their environment through the filter of imperfect sensors (which notably have a limited field of view with minimal overlap specifically along the perpendicular) and so may be prone to mis-perception or perceptual aliasing. It makes sense then that the distribution of behavior for a fish approaching a wall would exhibit two modes corresponding to both

choices[1]. It is also clear that in this scenario the mean or median behaviors correspond to a perfectly unnatural behavior (moving directly towards the wall) that nevertheless minimizes the expected difference between predicted and observed behavior training examples, and as such is the "best" fixed estimate of behavior. If the goal is to produce realistic predictions of behaviors of this kind, it is clear that one cannot rely on fixed-estimate predictors.

Using density estimates and sampling to create executable models

An obvious alternative is for $\hat{f}$ to estimate the true density f directly, and then *sample* under $\hat{f}$. Since the benefit of this sampling is only evident in cases where the distribution of behaviors are not accurately modeled by Gaussian or similar distributions, and in general the form of f is not known *a priori*, it is reasonable to rely on non-parametric density estimation. The standard approach to non-parametric density estimation is kernel density estimation, which takes the following form (for univariate a)

$$\hat{f}_h(a) = \frac{1}{|D|h} \sum_{a_i \in D} K\left(\frac{a - a_i}{h}\right) \tag{3.5}$$

where K is a chosen kernel function, and h is the bandwidth parameter. In the general case, when a can be multivariate, it is sometimes more convenient to rewrite K as a function of two arguments and absorb the bandwidth parameter:

$$K_h(a, a_i) = \frac{1}{h} K\left(\frac{\|a - a_i\|}{h}\right)$$

Note that this formulation is independent of ϕ, which is not useful if the agents must vary their behavior as a function of their environment, but the *conditional* form does rely

[1] It may even be advantageous for the fish to exhibit higher variance of behavior in the context of fleeing in response to a perceived threat, since any highly predictable behavior could be exploited by a predator.

on ϕ:

$$\hat{f}_h(a \mid \phi(e)) = \frac{\sum_j K_{h_a}(a, a_j) K_{h_\phi}(\phi(e), \phi(e_j))}{\sum_i K_{h_\phi}(\phi(e), \phi(e_i))} \tag{3.6}$$

This has been called the conditional kernel density estimator, or the Nadaraya-Watson conditional density estimator due to its similarity to the non-parametric regression technique of the same name (De Gooijer and Zerom 2003; Hyndman, Bashtannyk, and Grunwald 1996).

It is possible to efficiently approximate samples from this distribution if the kernel vanishes rapidly as distance increases and has small bandwidth: by using k-Nearest Neighbors to obtain the k tuples whose $\phi(e_i)$ values are closest to $\phi(e)$. These nearest neighbors will have the greatest effect on $\hat{f}$, and if k is chosen appropriately, the contribution of points further away would be negligible. The consistency of kernel density estimators is usually given asymptotically, and relies on the bandwidth parameter shrinking to zero as the number of data points grows to infinity. Many bandwidth selection programs (Holmes, Alexander Gray, and Isbell 2010; Holmes, Alex Gray, and Isbell 2007; Liu, Lafferty, and Wasserman 2007) mirror this by scaling h down as the number of samples increases. This suggests a further efficiency in the approximation by selecting uniformly from the nearest neighbors. This approximates drawing from a conditional kernel density estimator with the following kernels

$$K_{h_a}(a, a_i) = \delta(a, a_i), \qquad K_{h_\phi}(\phi(e), \phi(e_i)) = \begin{cases} \frac{1}{c} & s_i \in N_k(e) \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

where δ is the Dirac delta function, $N_k(e)$ is the set of k-nearest neighbors to e in the dataset D, and $c \propto \max(\{\|\phi(e) - \phi(e_j)\| : e_j \in N_k(e)\})$ is a constant proportional to the distance from e to its furthest neighbor. The number of distance comparisons performed which computing the nearest neighbors of a query point can be improved from the naive

linear complexity by using a spatial data structure, such as a kd-Tree, reducing the average complexity to approximately logarithmicMuja and Lowe 2014.

In addition to being faster than rejection sampling under (3.6), it has the added benefit of not requiring a bandwidth parameter, the tuning of which is a problem of considerable complexity in itself. By contrast, the k parameter is a positive integer that is relatively straightforward to tune using cross validation.

An executable model that produced outputs by sampling from among the k-nearest neighbors would be expected to produce a distribution of actions that matches the training examples, but it is less clear how well such a model would fair in terms of predictive performance, like single-step RMSE. The next section presents empirical results which quantify both predictive performance and distributional similarity for a neighborhood sampling kNN model and compare it with two models which optimize solely for predictive performance.

Experiments in learning models of synthetic and real fish schooling

The following experiments explore how fixed-estimate predictors fail to reproduce an appropriate distribution of behavior when modeling stochastic behaviors, and show how sampling predictors can address these issues. Further details on the experimental design and methodology is described in Hrolenok, Boots, and T. H. Balch 2017.

Each experiment follows the same workflow,

1. Construct a simulation of a group of agents with homogeneous behavior, deterministic or stochastic.

2. Run the simulation and collect tracks.

3. Construct executable models of the behavior using Linear Regression, kNN regression, and kNN sampling.

4. Run the learned models in simulation and collect tracks.

Table 3.1: Description of features σ_i.

Component	$v_{i,j}$	σ_i
$v_{\text{sep},j}$	Away from agent j $(-(x_j - x))$	Near, 1-2 body lengths (0.1)
$v_{\text{ori},j}$	Heading of agent j (θ_j)	Somewhat near, 2-3 body lengths (0.2)
$v_{\text{coh},j}$	Towards agent j $((x_j - x))$	Majority of school (1.0)
$v_{\text{obs},j}$	Away from obstacle j	Short, within 1 body length (0.05)

5. Compute the error metrics, distributional similarity and predictive performance.

The summary results of these experiments is that the sampling based predictor performs on-par with or slightly worse than fixed-estimate predictors in terms of predictive performance, but significantly outperforms fixed-estimate predictors in terms of distributional similarity.

Constructing an executable model using a trained predictor is straightforward: the output of the executable model is just the predicted output for the given feature values, which are in turn computed from the given environmental state. Two measures of predictive performance are considered: the single-step loss or RMSE across a hold-out set of $(\phi(e_i), a_i)$ tuples from the generating behavior, and the end-point error which is calculated as the distance between the final pose of a hold-out trajectory and the final pose of a trajectory generated by the learned model when initialized to the same configuration. Distributional similarity is measured by collecting tracks from both the synthetic and learned models and comparing their histograms.

Motivated by the earlier example involving fish, and the overarching goal of learning complex multiagent behaviors of physical systems, the synthetic behavior which generates the behavior consists of two models (deterministic, and stochastic) of schooling behavior for a small number of fish confined to a shallow tank.

Table 3.2: Feature weights W for the deterministic behavior. This table contains the ground-truth parameters for the generating behavior. Compared with Table 3.3, it is clear that it is possible to recover the parameters of the generating behavior by training on samples taken from tracks in the absence of noise. This result is not unexpected, but it provides evidence that the issues raised in subsequent experiments are not due to the difficulty of the learning problem.

Generating $f(\phi(s))$	$\dot{x}$	$\dot{y}$	$\dot{\theta}$
ϕ_{sep_x}	-1.0	0.0	0.0
ϕ_{sep_y}	0.0	0.0	-20.0
ϕ_{ori_x}	0.0	0.0	0.0
ϕ_{ori_y}	0.0	0.0	0.1
ϕ_{coh_x}	0.0	0.0	0.0
ϕ_{coh_y}	0.0	0.0	0.8
ϕ_{obs_x}	-1.0	0.0	0.0
ϕ_{obs_y}	0.0	0.0	-40.0
bias	0.0125	0.0	0.0

Table 3.3: Feature weights W for the deterministic behavior. This table contains the parameters recovered by linear regression. The recovered parameters nearly match the generating parameters (listed in Table 3.2), with the Frobenius norm of their difference being less than 0.894. This indicates the learning problem is not too difficult for linear regression to solve, as expected.

Recovered $\hat{f}(\phi(s))$	$\dot{x}$	$\dot{y}$	$\dot{\theta}$
ϕ_{sep_x}	-0.9998	0.0	0.0015
ϕ_{sep_y}	-5.1436×10^{-4}	0.0	-19.9995
ϕ_{ori_x}	-1.3071×10^{-5}	0.0	-9.3309×10^{-6}
ϕ_{ori_y}	1.1432×10^{-7}	0.0	0.0998
ϕ_{coh_x}	-3.3333×10^{-6}	0.0	2.8271×10^{-4}
ϕ_{coh_y}	3.1213×10^{-5}	0.0	0.8004
ϕ_{obs_x}	-0.9753	0.0	-0.3991
ϕ_{obs_y}	4.6335×10^{-4}	0.0	-38.7059
bias	0.01250	0.0	-3.5565×10^{-5}

Results for synthetic-deterministic schooling behavior

The deterministic model is similar to the well known Boids (Reynolds 1987) model. This model produces a realistic schooling behavior with each agent following a set of local rules that correspond to a linear combination of feature vectors. By specifically choosing a linear model the focus is on the differences between deterministic and stochastic behaviors, and shows that any limitations of a linear regression based predictor are not due to the expressiveness of the model or the complexity of the underlying behavior.

In this model, the behavior space ranges over desired velocities $b = (\dot{x}, \dot{y}, \dot{\theta})$ (where $\dot{x}$ and $\dot{y}$ are relative to the current heading θ), and the features correspond to vectors that capture characteristics of the school in a local area near the agent, which are summarized in Table 3.1 in the second column. These vectors are distance-weighted averages with the following form:

$$\phi_i(s) = \frac{1}{n} \sum_{j \neq i} \exp \left\{ \frac{d_j^2(s)}{2\sigma_i^2} \right\} v_{i,j}(s) \tag{3.8}$$

where n is the number of agents in the school, $d_j(s)$ is the distance to agent j, and $v_{i,j}(s)$ is a unit directional vector unique to ϕ_i. We use four ϕ_i: a *separation* component, an *orientation* component, a *cohesion* component, and an *obstacle* avoidance component. By choosing appropriate σ_i, the sensitivity of each component can be tuned so that each component responds to agents or obstacles at different ranges. The range of each σ_i is reported in the second column of Table 3.1.

The actual behavior f is defined as a simple linear combination of these features:

$$f(\phi(s)) = \langle W, \phi(s) \rangle \tag{3.9}$$

where $\phi(s)$ is the concatenation of the four feature vectors described above, along with a constant term: $\phi(s) = [\phi_{\text{sep}}(s), \phi_{\text{ori}}(s), \phi_{\text{coh}}(s), \phi_{\text{obs}}(s), 1]$, and W is described in Table 3.2.

Both W and each σ_i were hand tuned to produce reasonable looking schooling behavior where agents avoided collisions with each other and obstacles and preferred to stay grouped and aligned when possible.

Given this model, it is straightforward to train a predictor $\hat{f}$ with a set of observations collected from a simulation running the model. These observations are recorded directly by the simulator, and since all of the features can be computed from the pose (x, y, θ) of all the agents at any given time, they can stand in for tracking information gathered from video of live animals, as is shown in a subsequent section.

Training a linear regression model from these samples works well, unsurprisingly, and examining the learned weights shows that the parameters of the generating model can be recovered quite closely (Table 3.3). Similarly, kNN with neighborhood averaging (denoted by kNN-Reg from here on) performs well, while kNN with neighborhood sampling (kNN-Sample) performs slightly worse. The first three rows in Table 3.4 show the performance of all three methods in terms of average prediction error (RMSE) and average end-point error in the first two columns. For this simple model, this result is as expected, since the benefit of sampling from the predicted distribution doesn't come in to play for deterministic models and comes at the cost of increasing the variance of the error of the prediction.

Results for synthetic-stochastic schooling behavior

A slight variation of the model in the previous section which incorporates an element of randomness into the behavior suffices to illustrate the benefit of sampling based predictors. Starting with the same ϕ_i, and the same linear combination of these weights, a random variable sampled from an exponential distribution with mean $\beta = 0.05$ is added to the desired forward velocity ($\dot{x}$). Since this random variable is non-negative, the bias term in W is adjusted down to 0.00625 so that the mean forward velocity would remain approximately the same. This generates a behavior similar at a high level to what was generated before, but where the agents exhibit some unpredictable variation in forward speed. Since this

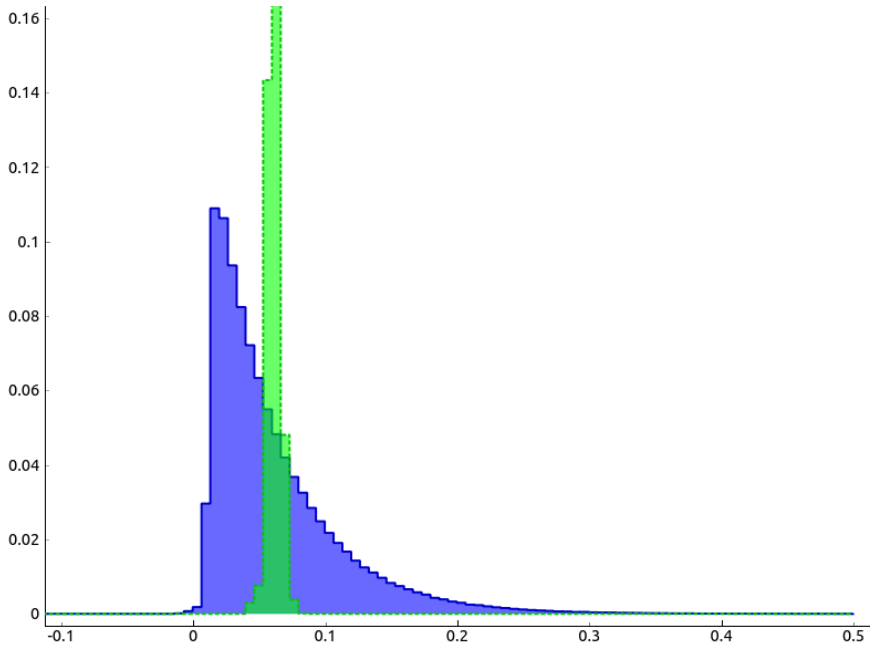

Figure 3.2: Distribution of agent x-velocity for linear regression in green (dotted line), versus generating agents in blue (solid line) for the stochastic simulated behavior. Note that while the means are similar in both, the distributions are significantly different.

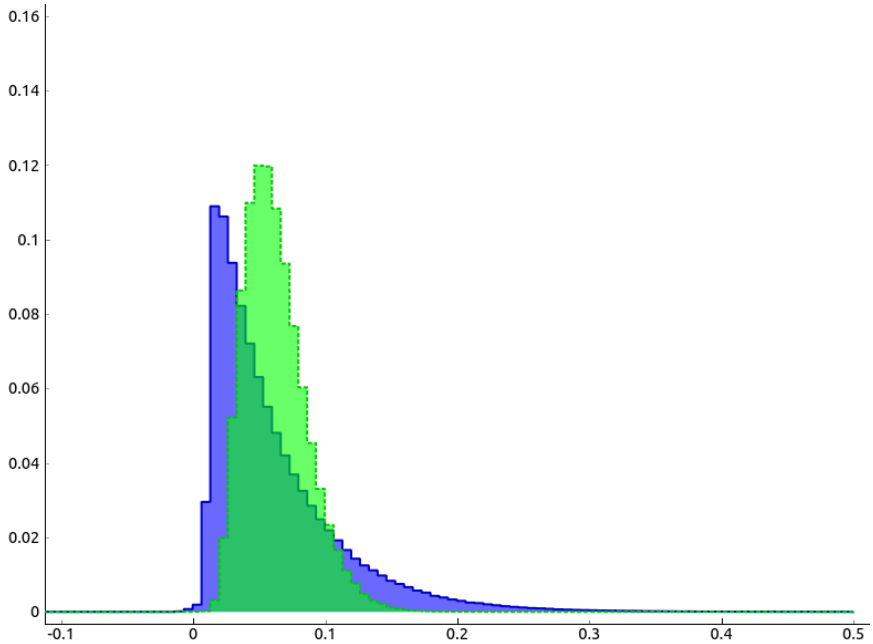

Figure 3.3: Distribution of agent x-velocity for kNN-Reg learned agents in green (dotted line), versus generating agents in blue (solid line) for the stochastic simulated behavior. Similar to 3.2 the mean of x-velocity is similar, but the distributions do not quite match. Specifically, note that the mode of the kNN-Reg distribution is closer to its mean and median, and its tails taper more rapidly.

Table 3.4: Performance of linear regression, kNN-Reg, and kNN-Sample. Reported values are mean and standard deviation for 10-fold cross validation. Lower is better for all three error metrics. The results show that while fixed-estimate predictors like linear regression and kNN-Reg perform best in terms of RMSE or end-point error, kNN-Sample is best in terms of K-L divergence. Additionally, kNN-Sample's performance is not substantially worse in terms of end-point error.

Deterministic

Predictor	RMSE	end-point	K-L divergence
Lin-Reg	$\mathbf{0.0356 \pm 0.0050}$	$\mathbf{0.0023 \pm 0.0006}$	—
kNN-Reg	5.9675 ± 0.3665	0.0070 ± 0.0021	—
kNN-Sample	4.4097 ± 0.3328	0.0069 ± 0.0020	—

Stochastic

Predictor	RMSE	end-point	K-L divergence
Lin-Reg	$\mathbf{6.1099 \pm 0.0361}$	$\mathbf{0.0067 \pm 0.0009}$	482.6036 ± 0.7310
kNN-Reg	13.5425 ± 0.4549	0.0150 ± 0.0017	51.1153 ± 12.3740
kNN Sample	15.1933 ± 0.4574	0.0154 ± 0.0014	$\mathbf{0.0465 \pm 0.0282}$

Live Fish

Predictor	RMSE	end-point	K-L divergence
Lin-Reg	$\mathbf{308.6853 \pm 22.4447}$	0.2164 ± 0.0148	1.5865 ± 0.1894
kNN-Reg	375.9885 ± 24.2010	$\mathbf{0.1877 \pm 0.0156}$	1.0020 ± 0.5021
kNN Sample	570.9620 ± 29.3125	0.2136 ± 0.0160	$\mathbf{0.2957 \pm 0.2497}$

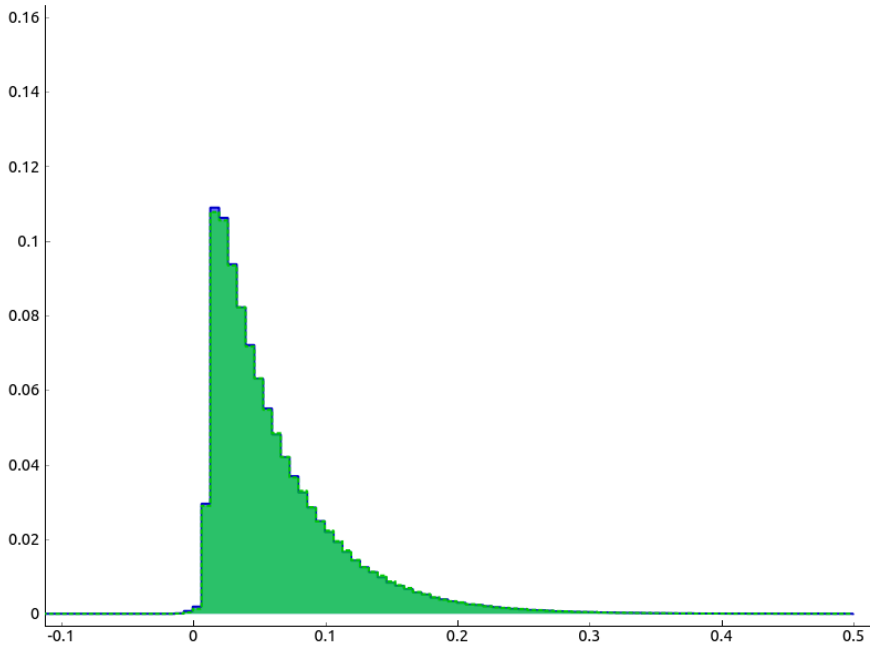

Figure 3.4: Distribution of agent x-velocity for kNN-Sample agents in green, versus generating agents in blue for the stochastic simulated behavior. Using this learned model, the distribution is nearly identical for both groups of agents.

variation is non-symmetric, the expected result is that the fixed-estimate predictors LinReg and kNN-Reg should produce behavior distributions that are significantly different from the generating behavior.

In addition to RMSE and end-point error a third performance metric is needed to measure the effect of this inherent unpredictability, namely the Kullback-Leibler divergence which is applied to histograms of forward velocity under the generating behavior and the predictor behavior. The third column in Table 3.4 represents the degree of difference between the two distributions where a K-L divergence of zero means the two distributions match exactly. Figures 3.2, 3.3, and 3.4 show the histograms graphically for a more qualitative comparison. As these figures show, the sampling based predictor produces a distribution of behavior that more closely matches the generating behavior, while the fixed estimate predictors do not, even though the three predictors perform similarly in terms of end-point and RMSE error.

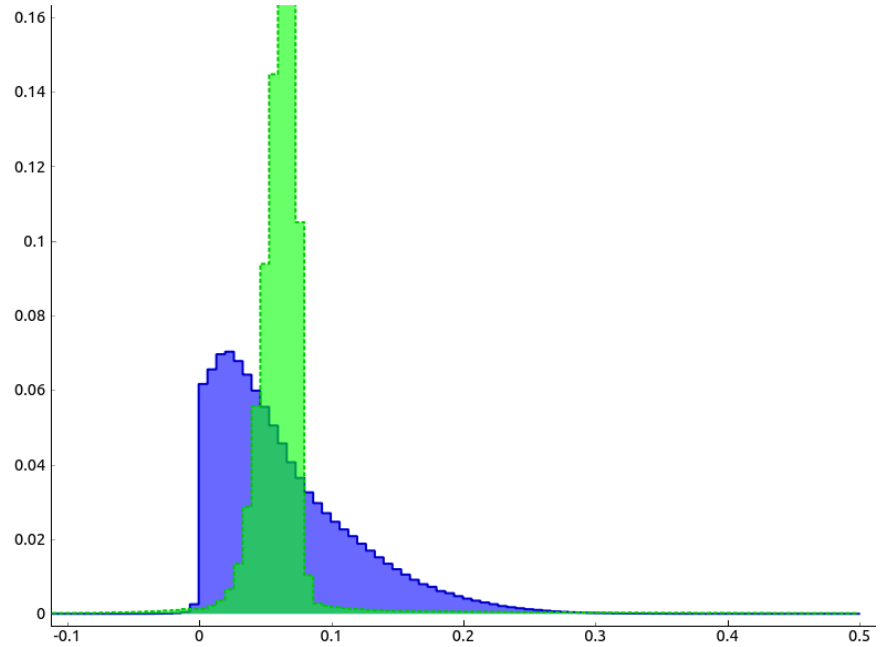

Figure 3.5: Distribution of agent x-velocity for linear regression in green (dotted line), versus actual fish in blue (solid line) for the data collected from real fish. Similar to the case with the stochastic simulated behavior, linear regression does a better job of matching the mean value than th shape of the distribution.

Results for real fish schooling behavior

In the previous two experiments, the training data was collected from a simulation without noise, and for which the ground truth parameters are known. This setup is useful for illustrative purposes, but unrealistic. This section explores a more interesting domain that motivated the choice of behaviors and examples in previous sections. Given pose information (x, y, θ) from video of a small school of *Notemigonus crysoleucas* in a shallow tank[2] it is possible to apply the same prediction methods and performance metrics to create executable models of real fish schooling behavior. The selection of ϕ_i for this problem is by no means straightforward, as will be discussed further in chapter 5, but as the focus of this chapter is on comparing learning methods with similar expressive power, a naive set of features is sufficient.

[2]Data was originally collected in Katz et al. 2011, which details the specifics of the environment, and the tracking method used.

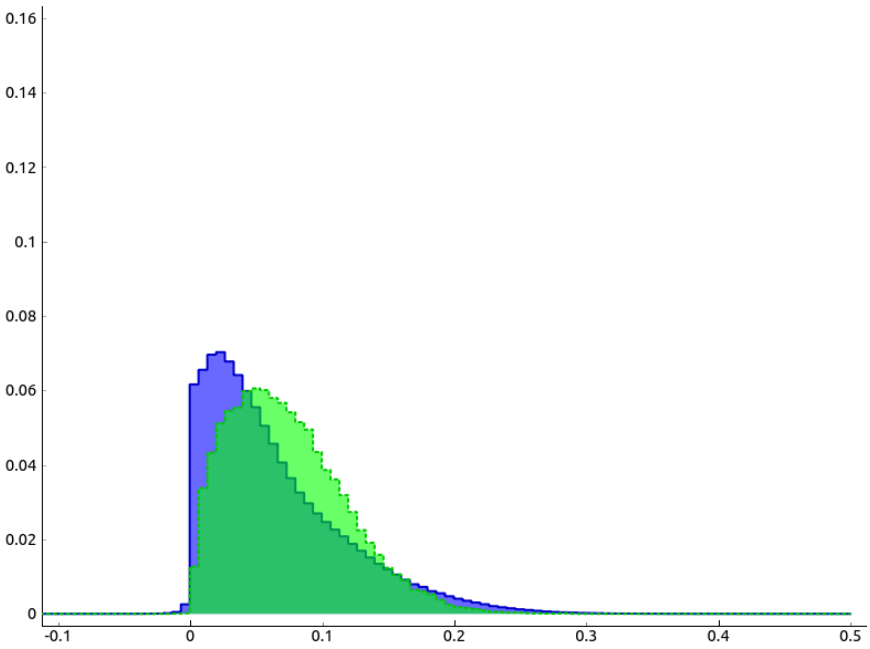

Figure 3.6: Distribution of agent x-velocity for kNN-Reg in green, versus actual fish in blue for the data collected from real fish. As before, the tails and the mode of the distributions do not match.

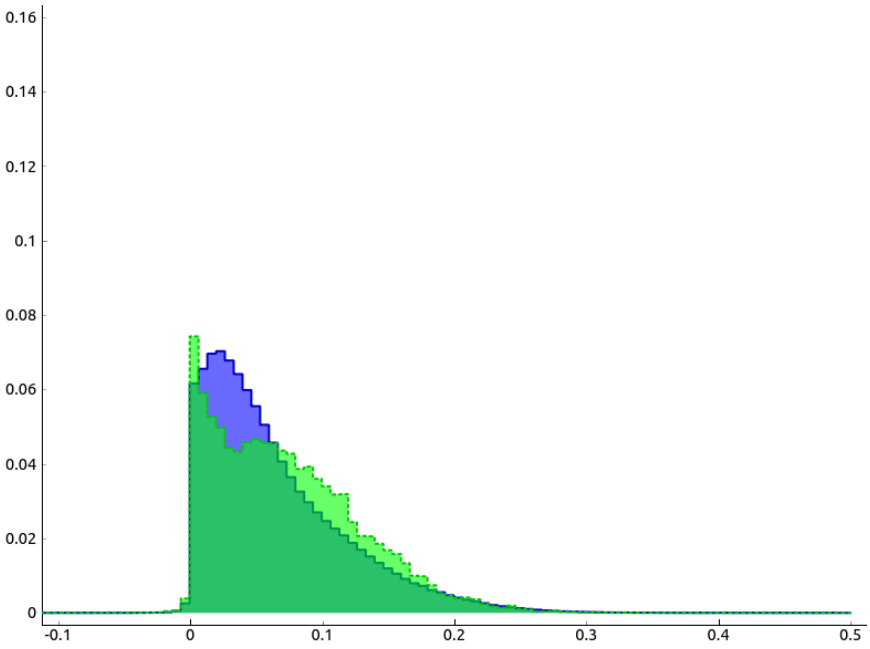

Figure 3.7: Distribution of agent x-velocity for kNN-Sample in green, versus actual fish in blue for the data collected from real fish. While not as stark as in the stochastic simulated behavior, the distributions are still more closely matched.

Such a set of features includes those ϕ_i described earlier, as well as a few additional features that were found to improve the performance of *all* predictors, with the intuition that these features provide reasonable coverage of the information that is relevant to schooling behavior for an individual agent, specifically the mean (ϕ_μ), variance (ϕ_σ), and maximum magnitude (ϕ_{max}) of the forward velocity of all other fish in the school.

For both kNN methods, the selection of k was handled by 10-fold cross validation on a parameter sweep from $k \in \{1 \dots 100\}$. For both methods the maximum performance was found at $k = 5$. The entire dataset was taken from a one hour video shot at 30 frames per second, with 30 fish visible in each frame, roughly 1.4 million data points after filtering.

The performance of linear regression, kNN-Reg, and kNN-Sample are given numerically in Table 3.4, and graphically in Figures 3.5, 3.6, and 3.7. Note that the distribution of forward velocity for the real fish is an asymmetric Poisson-like curve, which is what motivated the stochastic model from the previous experiment. The form of the true behavior f for real fish is not known, and there is no strong evidence that it is linear, so it is not surprising that linear regression does not outperform non-parametric methods. As before, the increased variance in the expected difference between prediction and actual behavior for kNN-Sample leads to larger RMSE and end-point error than the fixed-estimate predictors, but a closer match between the distribution of predicted and actual behavior.

Limits in representations without behavioral state

This chapter has shown that the kNN-Sample method can perform comparably with other learning methods in terms of predictive performance while significantly improving the distributional similarity of the learned behavior to the generating behavior. In the synthetic example of Boids schooling, it was clear that the reason for this mismatch was not due to the complexity of the underlying behavior, since the underlying behavior was explicitly linear. However, it is not clear that real fish behave in a way that can be captured by such a simple model. Certainly there are some behaviors that are more naturally described in terms

of some internal state, as described in chapter 1. The next chapter covers learning models that incorporate the notion of state, building upon the kNN-Sample model presented in this chapter.

CHAPTER 4

LEARNING EXECUTABLE MODELS WITH BEHAVIORAL STATE

Chapter 3 presented kNN-Sample as a method for modeling stochastic behaviors. Being a non-parametric model this method presumably has the capacity to represent a wide variety of behaviors, but the restriction to stateless behaviors might exclude a broad subset of interest. This chapter presents a model which builds upon kNN-Sample that incorporates state in an intuitive way, and a method for learning this model from tracking data.

Modeling stateful behavior

The approach to modeling behavior with state taken in this chapter is to represent the behavior as a Finite State Automata, similar to the approach outlined by Yang et al. 2012. More specifically, the behavior of an agent can be described by a graph in which each node represents a state the agent can be in, and each edge represents a transition from one state to another, and the label for that edge corresponds to a boolean clause that must evaluate true for the agent to make that transition. Within a specific state, the agent's output is a direct function of its sensors. That is, the agent has no "memory" outside of the state structure. This graphical representation of behavior is quite similar to the ethograms that biologists use to describe observed behavior Schleidt et al. 1984.

Figure 4.1 gives an example of how one might graphically represent a simple foraging

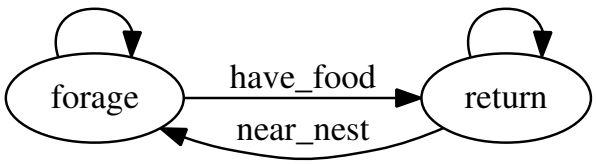

Figure 4.1: A simple foraging behavior

behavior for an ant. The ant remains in the foraging and returning states until it finds food or is near the nest respectively. While in the foraging behavior, the ant might head towards the nearest food item within its sensor range, and while returning it might head in the direction of the nest. In both of these states the ant might try to avoid walls and other ants. This can be naturally extended to include a probabilistic transition function rather than a deterministic one.

A stateful behavior can be formally defined under this model as a triple

$$\langle \pi, A_{s,s'}(\tau), \{f_s(a \mid \phi(e)) \mid s = 1 \ldots N\} \rangle$$

where π_s is the probability of starting in state s, $A_{s,s'}(\tau)$ is the probability of transitioning from state s to s' given the logical clause τ, and $f_s(a \mid \phi(e))$ is the low-level behavior which gives the probability density of a specific response to a given set of perceptions computed from the environmental state $\phi(e)$ for behavioral state s. Note the difference between *environmental* state e and *behavioral* state s. The behavioral state is internal to the agent, and functionally simply controls which low-level behavior f_s the agent is currently following. The environmental state is external to the agent, and mediated by the sensor model ϕ. Generally, for the remainder of this chapter, state without qualification will be used to refer to behavioral state.

A specific behavior uses a finite number of boolean variables, k, in the clauses that control transitions between the N states, so A can be represented as a $N \times N \times 2^k$ table, and τ is the index of a specific configuration of the boolean variables. The intent of these boolean variables is that they should summarize aspects of the environment that are important in determining whether the behavior should switch behavioral states, and so it is reasonable that they should be computable from the environmental state. Each boolean variable is a mapping $q_j : \mathcal{E} \rightarrow \{0, 1\}$, and τ then is also a function of e: the integer corresponding to

the binary representation of the concatenation of all such variables

$$\tau(e) = \langle q_k(e), q_{k-1}(e), \ldots, q_1(e) \rangle \in \mathbb{N}$$

The sensor model $\phi(e)$ is a vector-valued function in $\mathbb{R}^d$, and as mentioned earlier f_s is a function strictly of $\phi(e)$ which without loss of generality will be vector valued (so $\mathcal{A} = \mathbb{R}^m$).

To give the full description of the behavior in terms of the formalism given in chapter 1

$$f(s_t, \phi(e_t)) = \langle a_t \sim f_{s_t}(\cdot \mid \phi(e_t)), s_{t+1} \sim A_{s_t,\cdot}(\tau(e_t)) \rangle \tag{4.1}$$

where $A_{i,\cdot}(\tau)$ is one row of $A_{i,j}(\tau)$ corresponding to the transition probabilities for state i given boolean variables τ.

The following definitions provide convenient labels for the remaining discussion:

- τ_t: the index corresponding to which boolean variables are true at time t, $\tau_t = \tau(e_t)$.
- $\mathbf{y}_t$: the value of the sensors at time t, $\mathbf{y}_t = \phi(e_t)$.
- $\mathbf{z}_t$: the agent's action at time t, $\mathbf{z}_t = f(s_t, \phi(e_t))$.
- s_t: the internal state of the agent at time t.

If one considers sequences of triples consisting of inputs, binary switches, and outputs $\xi_t = \langle \mathbf{y}_t, \tau_t, \mathbf{z}_t \rangle, t = 1 \ldots T$, of an agent following behaviors of this form, one can construct a graphical model which describes the probability of any particular sequence along with the unobserved state variables, as shown in Figure 4.2. Specifically, the action $\mathbf{z}_t$ ("turn θ radians left", "go forward") that the agent takes at time t depends on its sensors $\mathbf{y}_t$ (vectors in the direction of the nest/food), and its current state s_t ("forage", "return"). The state at the next time step s_{t+1} is dependent on both the previous state s_t, and state of the boolean variables τ ("have_food" and "near_nest"). This model makes it straightforward to recast

the problem of learning a behavior into finding the set of parameters which maximizes the likelihood of the data under those parameters, which in standard HMMs can be solved using Expectation Maximization, also known as the Baum-Welch algorithm for HMMs due to Baum et al. 1970. Section 4.2 introduces an iterative parameter updating procedure similar to the Baum-Welch update equations.

This is a natural extension of the classic HMM model which makes explicit the dependence relationships specific to this domain which derive from the fact that an agent is *reacting* to its environment. The IOHMM presented by Bengio and Frasconi 1995 also exploited this relationship, but that model made no distinction between the parts of the input that influenced state transitions and the parts that effected the state output (Figure 4.3). This modification splits the input variable in two, and separates the variables that are important to state switching from those that are important to how the agent responds in a given state. The BIOHMM model is not any more expressive than an IOHMM, but it allows greater freedom in the choice of how the low-level behaviors f_s are modeled, and in the feature design.

Learning stateful models

Learning a behavior $\Theta = \langle \pi, A_{s,s'}(\tau), \{f_s(a \mid \phi(e))\} \rangle$ given a sequence of observations $\Xi = \{\xi_t = \langle \mathbf{y}_t, \tau_t, \mathbf{z}_t \rangle \mid t = 1 \dots T\}$ can be accomplished through Expectation Maximization in a way similar to how Baum-Welch solves the analogous problem for HMMs.

First, the likelihood for a given sequence in the complete data case can be written as follows:

$$\begin{aligned} L(\Xi, S \mid \Theta) &= P(s_0) \cdot \prod_t P(s_t, s_{t+1} \mid \tau_t) \cdot L(\mathbf{z} \mid f_{s_{t+1}}, \mathbf{y}_{t+1}) \\ &= \pi_{s_0} \cdot \prod_t A_{s_t, s_{t+1}}(\tau_t) \cdot f_{s_{t+1}}(\mathbf{z}_t \mid \mathbf{y}_{t+1}) \end{aligned}$$

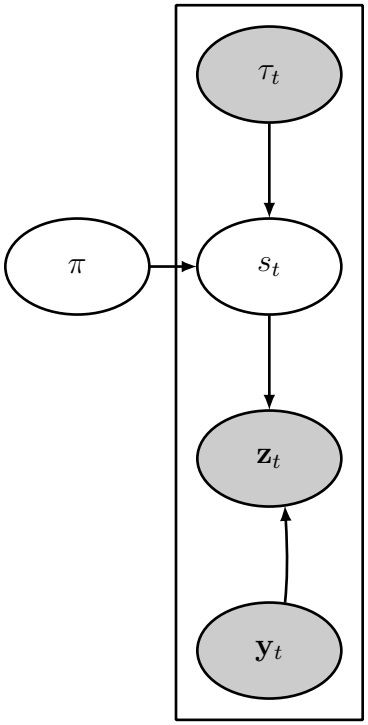

Figure 4.2: A Binary-switched IOHMM (BIOHMM). Observed variables are shaded

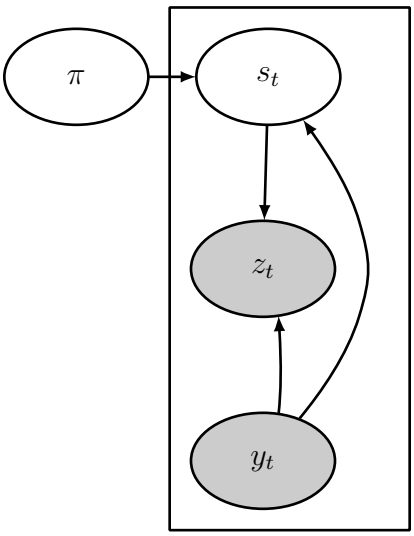

Figure 4.3: An IOHMM. Observed variables are shaded

where $L(\mathbf{z} \mid f_s, \mathbf{y}) = f_s(\mathbf{z} \mid \mathbf{y})$ is the likelihood that the output function for state s produced output $\mathbf{z}$ for an input $\mathbf{y}$.

The likelihood of a specific (complete data) sequence is just the product of the probability of the initial state, the probability of each state transition, and the likelihood of each observed output. The differences between a BIOHMM and a standard HMM only come in to play with variables that are observed in the training data, so the changes in forward-backward and the Baum-Welch updates are intuitive, and involve conditioning on the input and switching variables[1]. The forward variable, α, is computed by:

$$\begin{aligned}
\alpha_t(s) &= P(s_t = s \mid \xi_{1:t}) \\
\alpha_1(s) &= \pi_s \cdot f_s(\mathbf{z}_0 \mid \mathbf{y}_0) \\
\alpha_{t+1}(s') &= \left[\sum_s \alpha_t(s) A_{s,s'}(\tau_t) \right] f_{s'}(\mathbf{z}_{t+1} \mid \mathbf{y}_{t+1})
\end{aligned}$$

The backward variable, β:

$$\begin{aligned}
\beta_t(s) &= P(s_t = s \mid \xi_{t:T}) \\
\beta_T(s) &= 1 \\
\beta_t(s) &= \sum_{s'} A_{s,s'}(\tau_t) \cdot f_{s'}(\mathbf{z}_{t+1} \mid \mathbf{y}_{t+1}) \cdot \beta_{t+1}(s')
\end{aligned}$$

The smoothed posterior marginal, γ:

$$\begin{aligned}
\gamma_t(s) &= P(s_t = s \mid \xi_{1:T}) \\
&= \frac{\alpha_t(s)\beta_t(s)}{\sum_{s'} \alpha_t(s')\beta_t(s')}
\end{aligned} \tag{4.2}$$

It is convenient to define an intermediate variable σ when computing the update to the

[1]This section uses the standard notation for EM on HMMs. For definitions and details on the original Baum-Welch readers are referred to Rabiner 1989

transition function:

$$\begin{aligned}\sigma_t(s, s') &= P(s_t = s, s_{t+1} = s' \mid \xi_{1:T}) \\ &= \frac{\alpha_t(s) A_{s,s'}(\tau_t) f_{s'}(\mathbf{z}_{t+1} \mid \mathbf{y}_{t+1}) \beta_{t+1}(s')}{\sum_s \sum_{s'} \alpha_t(s) A_{s,s'}(\tau_t) f_{s'}(\mathbf{z}_{t+1} \mid \mathbf{y}_{t+1}) \beta_{t+1}(s')}\end{aligned} \tag{4.3}$$

There is no difference between BIOHMMs and HMMs that effects the initial state distribution π, which leaves its update equation unchanged:

$$\pi'_s = \gamma_0(s) \tag{4.4}$$

Since the transition matrix is now conditioned on the binary switching variable, the update must be modified to only account for transitions under a specific τ:

$$A'_{s,s'}(\tau) = \frac{\sum_t \sigma_t(s, s') \delta_{\tau_t}(\tau)}{\sum_t \gamma_t(s) \delta_{\tau_t}(\tau)} \tag{4.5}$$

where $\delta_{\tau_t}(\tau)$ is the indicator function for switching variable τ at time t.

Output function

One of the key distinctions made between the BIOHMM model and the IOHMM model is how the output function is modeled. In the original IOHMM a neural network was used to model the output function, and back propagation was used to calculate the update as part of the EM step. In contrast, so far the form of the output function has not been constricted. In order to compute the updates and variables described above, some estimate of the probability that the output function for a given state produced a given output for a given input is necessary.

Following from chapter 3, an obvious choice is conditional kernel density estimation.

When executing the behavior an efficient sampling method is desirable, and it makes sense to use kNN-Sample to approximate CKDE samples. The kernels shown in 3.7 for which kNN-Sample is an approximation are not suitable for calculating the update equations for α,β,γ, and σ. Since the kernels are impulses at the training points, and zero elsewhere, $f_s(\mathbf{z}_t \mid \mathbf{y}_y)$ would be an indicator function for whether a given data point $(\mathbf{y}_t, \mathbf{z}_t)$ belonged to a given state s. The solution is to consider a soft assignment of data points to states. Instead of using equation 3.6 with the kernels described in 3.7, the likelihood f_s for each state uses Gaussian kernels which incorporate the soft assignment weights

$$f_s(\mathbf{z} \mid \mathbf{y}) = \frac{\sum_j w_j^{(s)} \cdot K(\mathbf{z}, \mathbf{z}_j) K(\mathbf{y}, \mathbf{y}_j)}{\sum_i w_i^{(s)} K(\mathbf{y}, \mathbf{y}_i)} \tag{4.6}$$

where $w_i^{(s)}$ is the soft assignment of data point i to state s, and

$$K(\mathbf{z}, \mathbf{z}') = \frac{1}{\sqrt{(2\pi\sigma)^2}} \exp\left\{ \frac{-\|\mathbf{z} - \mathbf{z}'\|^2}{2\sigma^2} \right\}$$

The weights are initially randomly assigned fully to one of the possible states, with the other states set to zero. Then, in each EM iteration the weights are updated according to the likelihood of the behavioral state active at the time the data point was recorded:

$$w_t^{(s)\prime} = \frac{\gamma_t(s)}{\sum_{t'} \gamma_{t'}(s)} \tag{4.7}$$

This updates the weight of the t^{th} sample point for output density estimate f_s for state s to be proportional to the probability of being in that state when that output was generated. In the special case where $\sum_{t'} \gamma_{t'}(s) = 0$ the weight is set $w_t^i = 0$ in equation 4.7 since, by construction, this means that $\gamma_t(i) = 0$. Similarly, because all weights are non-negative, if $\sum_i w_i = 0$ in equation 4.6, then the output density estimate is set to $f_s(\mathbf{x}) = 0$. This ensures that the output probability density is bounded.

Given equations 4.4, 4.5, and 4.7, EM can iteratively compute an improved Θ. Given a

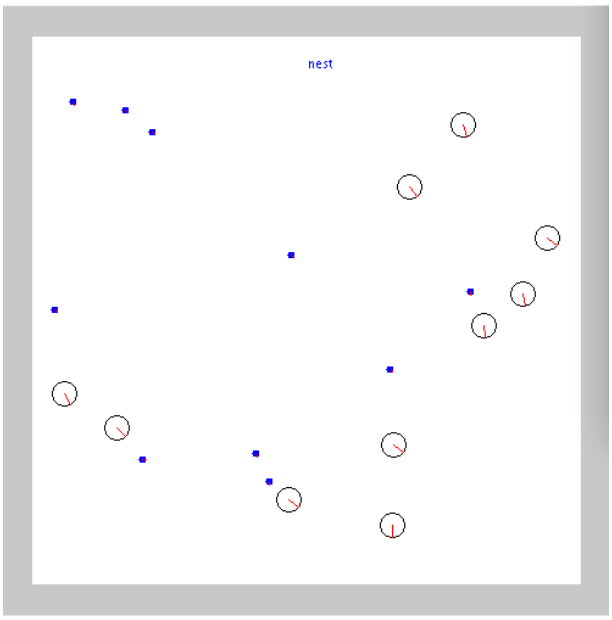

Figure 4.4: The simulation environment. Ants are the larger transparent circles, food items are the smaller filled circles

specific Θ, the transition function A and initial state distribution π can be used along with kNN-Sample for sampling the output function as an executable behavior.

Experiments in learning models of synthetic foraging and real ant behavior

The following experiments focus on the performance of behavior generated by a BIOHMM model learned using the methods described in the previous section.

When assessing the performance of an executable model of behavior, it is important that an agent acting under a learned behavior actually *behaves* like the animal that generated the training data. For synthetic behaviors, it is possible to parameterize the generating behavior in such a way that it should be possible to recover those parameters. To this end, the experiment described next explores the ability of a BIOHMM model to recover the parameters of a simulated group of foraging ants. The same process will be applied later to tracks of real ants, although the analysis is necessarily more indirect and qualitative.

The foraging simulation consisted of 10 ants and 10 stationary food items in a rectan-

gular arena. The ants were programmed to follow the behavior shown in Figure 4.1. While in the foraging state, the ant would head towards the nearest food item within range, and wander randomly until it came within range of a food item. While returning, the ant would head in the direction of the nest. If the ant got close enough to touch a food item in the foraging state, it would pick it up and transition to the return state. Similarly, when the ant returned to the nest while in the return state, it would drop any food it was carrying and transition back to the foraging state. In both states, the ant would try to avoid running into walls and other ants. Transitions in each of these cases were deterministic, and the ants always started in the foraging state. Sensor ranges were set to three body-lengths (25.8mm). Figure 4.4 is a screenshot of the simulation at initialization.

To collect training data, traces were recorded from the hand-coded behavior of each ant's output velocity ($\mathbf{z}$), the 4 sensor vectors ($\mathbf{y}$) — obstacle sensor, ant sensor, food sensor and direction-to-home sensor — and the two binary switching sensors "have_food" and "near_nest" (τ) at each simulation step. A model with two states was then trained using these traces as the observation sequences as described in section 4.2. The initial values for π and A were chosen randomly and each $\langle \mathbf{y}_t, \mathbf{z}_t \rangle$ was randomly assigned to one of the output functions. For the kernel density estimator a bandwidth of $\sigma = 1$ was used. After converging, the learned parameters were used to construct an executable behavior, and a new simulation was run with 10 food items and 10 ants using the learned behavior.

A random behavior was also implemented to serve as a baseline comparison, where the ants simply moved in a random direction at each time step. Whenever an ant moved over a food item it automatically picked it up unless the ant was already carrying food, and whenever it moved over the nest while carrying a food item it dropped the item off.

Results for synthetic foraging

The synthetic foraging behavior is simple enough that the learned parameters can be compared directly with the model that produced the training data. Figures 4.5, 4.6, 4.7, and

Table 4.1: Foraging transition function. HF and NN stand for the boolean variables have_food and near_nest.

	¬HF ¬NN		¬HF NN	
	RETURN	FORAGE	RETURN	FORAGE
RETURN	—	—	—	—
FORAGE	0.0	1.0	0.0	1.0

	HF ¬NN		HF NN	
	RETURN	FORAGE	RETURN	FORAGE
RETURN	1.0	0.0	0.0	1.0
FORAGE	—	—	—	—

Table 4.2: Aggregate statistics for hand-coded foraging.

	HAND-CODED	LEARNED	RANDOM
RETURN TIME	15.5 ± 1.1(S)	34.5 ± 3.9(S)	80.7 ± 8.5(S)
DISTANCE FROM NEAREST ANT	19.6 ± 0.02(MM)	14.36 ± 0.03(MM)	12.3 ± 0.03(MM)
DISTANCE FROM NEAREST WALL	17.6 ± 0.04(MM)	7.24 ± 0.02(MM)	5.5 ± 0.01(MM)
FOOD COLLECTED PER RUN	9.45 ± 0.26	6.25 ± 0.6	0.9 ± 0.35

4.8 shows how the transition function converges to the correct values given in Table 4.1. Figures 4.9 and 4.10 provide a visualization of a slice of the output function at iterations 1 and 40. The slice illustrated in this graph fixes the value of the direction-to-home sensor to directly in front of the ant, with the other sensors set to zero, and plots the probability density of the output angular velocity for the return state. Initially, the density is fairly uniform: no one turn is much more likely than any other. After several iterations, the ant has learned to preferentially move in the direction of the nest (zero angular velocity) when returning food.

The parameters were recovered correctly, but in the case of real ants these parameters are not accessible, so it is necessary to develop some quantitative measure of the agent's performance that can be computed for simulated and living agents in the same way that captures some aspect of the behavior that is interesting. In the case of this simulated foraging, the key behavioral components are

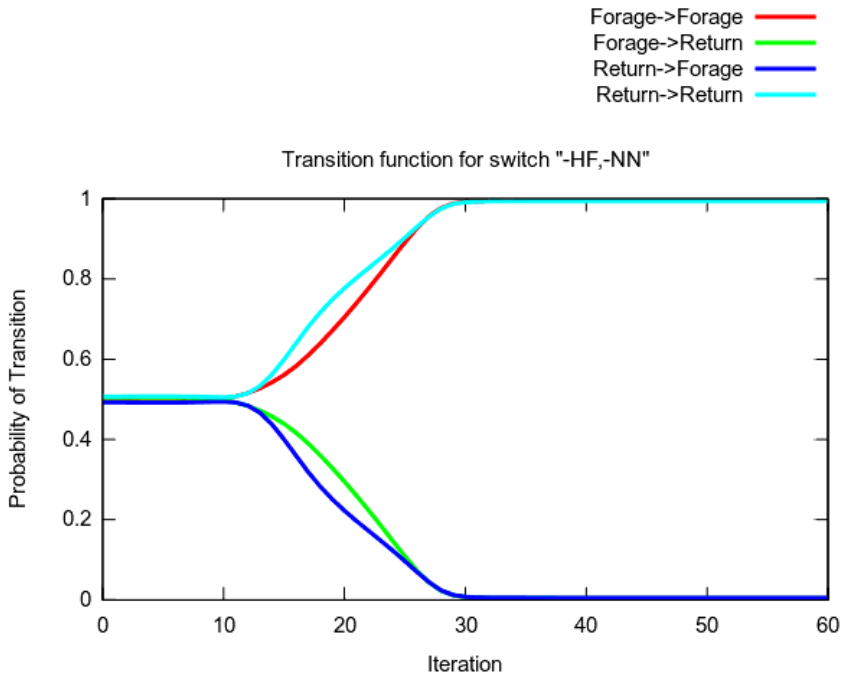

Figure 4.5: Evolution of the transition function parameters over time for switch values "have_food = false", "near_nest = false". Transitions that occur in the hand-coded model converge to their correct values given in Table 4.1, indicating that the model is able to learn the ground truth parameters from tracks.

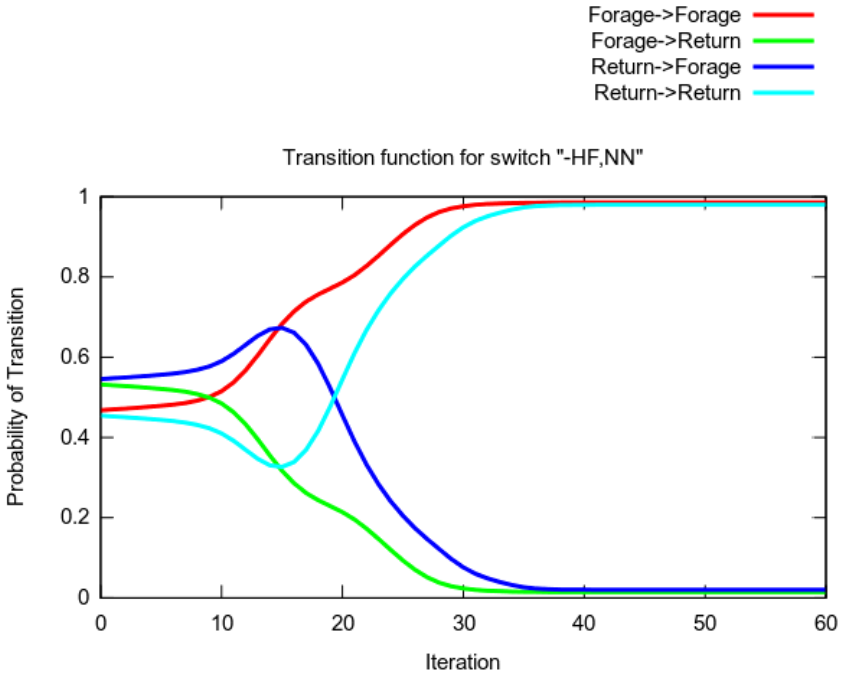

Figure 4.6: Evolution of the transition function parameters over time for switch values "have_food = false", "near_nest = true". Transitions that occur in the hand-coded model converge to their correct values given in Table 4.1, indicating that the model is able to learn the ground truth parameters from tracks.

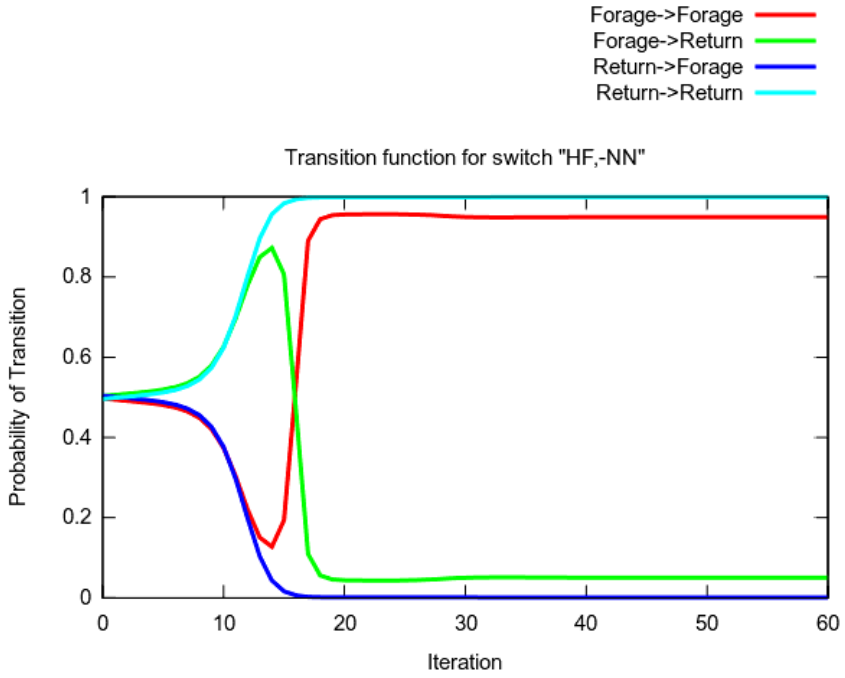

Figure 4.7: Evolution of the transition function parameters over time for switch values "have_food = true", "near_nest = false". Transitions that occur in the hand-coded model converge to their correct values given in Table 4.1, indicating that the model is able to learn the ground truth parameters from tracks.

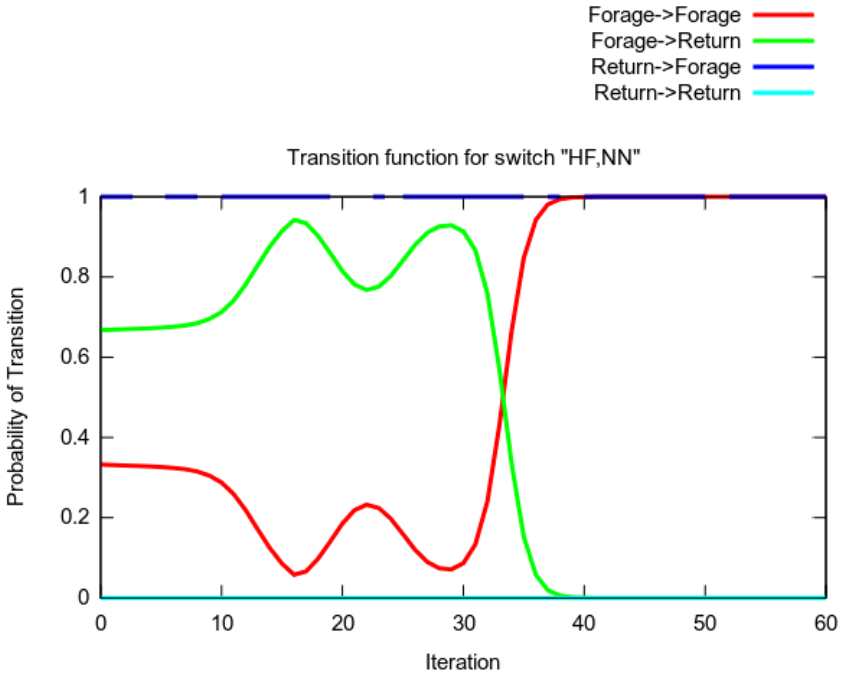

Figure 4.8: Evolution of the transition function parameters over time for switch values "have_food = true", "near_nest = true". Transitions that occur in the hand-coded model converge to their correct values given in Table 4.1, indicating that the model is able to learn the ground truth parameters from tracks.

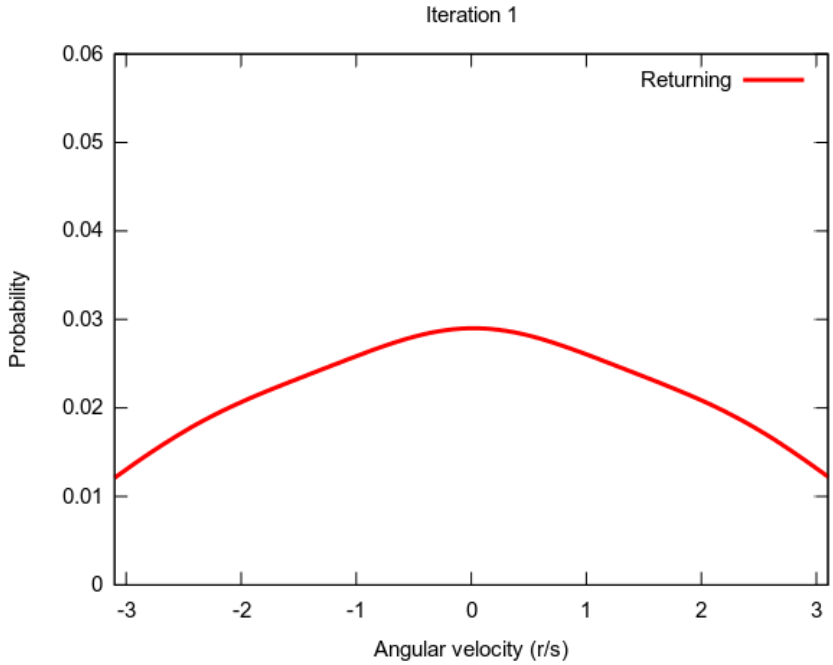

Figure 4.9: Evolution of the output function density. The graph is a slice of the output function for fixed sensor values, with the nest directly in front of the agent. Initially the output distribution has high variance, as the agent does not preferentially move towards the nest.

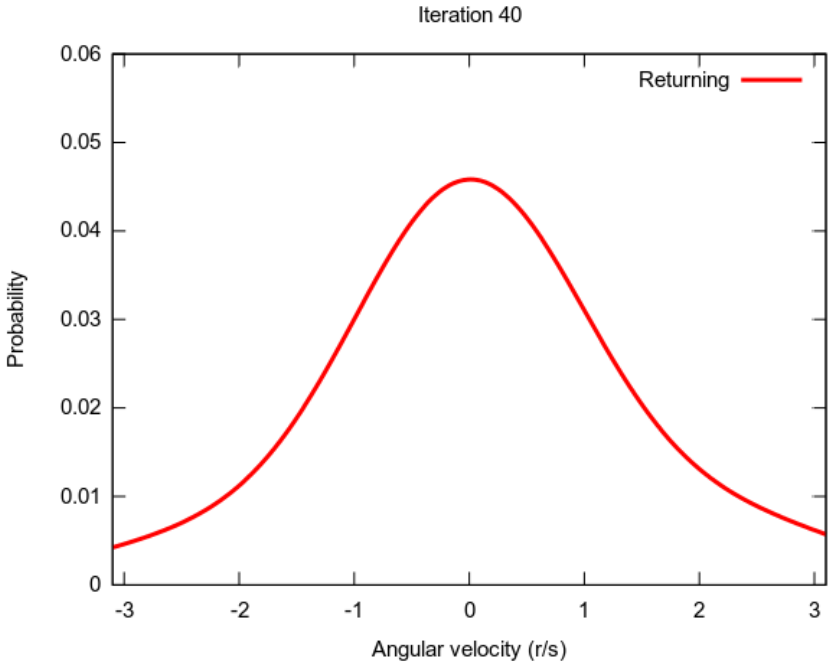

Figure 4.10: Evolution of the output function density. The graph is a slice of the output function for fixed sensor values, with the nest directly in front of the agent. After 40 iterations, the variance is much lower, indicating that the agent has learned to prefer moving towards the nest.

1. A preference for avoiding collisions with walls and other ants

2. A preference for seeking out and returning food items in the arena

so measures which relate to some aspect of the distance between an agent and things it should avoid, and how successful the agents are at collecting food would seem reasonable candidates.

Table 4.2 lists the aggregate data collected in the three simulations. Simulations were run for 120 simulated seconds (≈ 4000 steps), and were repeated 20 times; mean values are reported with 95% confidence intervals. RETURN TIME is the average time (in seconds) between when a food item is first picked up, and when it is dropped off near the nest. DISTANCE FROM NEAREST ANT is the average distance (in millimeters) between each ant and the nearest other ant within sensing range. DISTANCE FROM NEAREST WALL is the average distance between each ant and the nearest wall within sensing range. FOOD COLLECTED PER RUN is the average number of food items returned to the nest by all ants throughout the course of a single run.

These results illustrate that the executable behavior constructed from the learned parameters produces similar aggregate statistics to the original behavior. While the RETURN TIME is longer in the learned behavior, it is still much shorter than the time it takes a randomly wandering ant, which suggests the learned behavior is trying to head towards the nest after picking up a food item. Similarly, the fact that the average distance between nearby ants and obstacles is larger for the learned behavior than the random behavior suggests that the ant is attempting to avoid collisions. The FOOD COLLECTED PER RUN suggests that the learned behavior produces aggregate behavior that is very similar to the hand-coded foraging ants.

Results for live animals

All of the methods described earlier in this chapter can be applied to learning models of real animals. In this experiment, video was taken in the lab of a group of *Aphaenogaster*

Figure 4.11: One frame of a video of *A. cockerelli* in an empty arena

Table 4.3: Aggregate statistics for wandering ants.

	REAL	LEARNED
DISTANCE FROM NEAREST ANT	15.11 ± 0.73(MM)	17.5 ± 0.03(MM)
TIME SPENT NEAR WALL	3.1 ± 1.3(S)	2.9 ± 0.3(S)
TIME SPENT NEAR ANTS	1.6 ± 0.5(S)	1.3 ± 0.08(S)
TIME SPENT NEAR NEST	2.2 ± 1.5(S)	3.8 ± 0.7(S)

cockerelli, a type of desert ant that is studied for its complex cooperative behaviors, in an empty arena.

Figure 4.11 shows a frame from this video. The ants in the video were tracked using Multi-Iterative Closest Point tracking Feldman, Hybinette, and T. Balch 2012 which provides the x, y, and θ of each ant in each frame. These tracks were then processed to create egocentric ant sensor values (including binary switches) paired with changes in (x, y, θ), and combined into sequences.

This provided the training data for a two-state model of ant wandering behavior, which was then compared to a held-out set of tracks of the real ants for comparison purposes.

Table 4.3 lists several aggregate statistics of both the real and learned ants. These kinds of interaction statistics are useful in answering important questions about how the animals in question behave socially, such as whether ants have a preferred encounter rate D. M. Gordon, Paul, and Thorpe 1993. As there is no ground truth to compare with, it is difficult to judge how well the learned behavior is performing, although most of the metrics are similar.

Formalizing a quantitative measure of behavior similarity

As was highlighted at the end of the previous section, in order to validate whether a learned executable model of behavior has been successfully trained some notion of quantitative similarity is necessary. The aggregate statistics reported in those experiments provide a first approximation, but much more detail can be obtained by comparing the full distributions of these statistics instead of just their first moments, as was seen in chapter 3. In the next chapter this intuition is expanded, and a full formal treatment is presented for a quantitative measure of behavioral similarity based on distributional similarity.

CHAPTER 5

ASSESSING EXECUTABLE MODELS OF BEHAVIOR WITH BEHAVIORAL DIVERGENCE

At this point, the need for a quantitative metric of behavioral similarity other than predictive performance in the form of single-step RMSE is clear. This chapter presents such a metric based on distributional similarity called Behavioral Divergence, connects it to other quantitative measures and optimization objectives, and finally illustrates its use in an experimental setting to evaluate the quality of a learned model of fish schooling behavior.

Behavioral divergence

The distributional similarity objective described in the introduction (1.2) is written in terms of the distribution of behavior, but it is important to make this precise. Martin and Bateson 1993 defines behavior as "the actions and reactions of whole organisms," but neglects to mention the context in which these actions and reactions takes place. The working definition adopted in this thesis is that behavior is *the observable responses of a group of agents to their environment, including other agents*. This last component is especially important for the collective behaviors that the executable models described in this thesis are aimed to model, as the interaction among agents is a key component of collective behavior.

With this definition in hand, the next question is one of representation. What comprises an "observable response?" In order for the representations to be useful they must be computable, not only for agents running in simulation, but also for animals like ants or fish for

which only tracking data exists. Extending the notation from chapter 4, let

$$\rho_t = \langle e_t, (\phi^{(1)}(e_t), a_t^{(1)}), (\phi^{(2)}(e_t), a_t^{(2)}), \ldots (\phi^{(N)}(e_t), a^{(N)}) \rangle$$
$$\xi = \{\rho_t \mid t = 1 \ldots T\} \tag{5.1}$$

represent a set of tracks for a set of N agents. These tracks consist of the state of the environment (e_t), and the actions ($a_t^{(i)}$) and perceptions or sensors ($\phi^{(i)}(e_t)$) of each agent $i = 1 \ldots N$ for each timestep. In this instance e_t consists of everything that is computable given the sensing modality used to collect the tracks, which is typically far more than what an individual agent can sense as represented by ϕ.

It is tempting to consider ξ as the object of interest, but it quickly becomes computationally infeasible to work with since the dimensionality of $\xi \in \Xi$ grows with the number of time steps and $\rho \in \mathcal{R}$ with the number of agents. Consider instead a collection of functions of ρ, $m_i : \mathcal{R} \to \mathbb{R}$ that summarize the aspects of behavior that are relevant, and denote these functions as *behavioral measures*, and let M_i represent the distribution of m_i over time. When M_i is not known and must be estimated from samples, $\hat{M}_i(\xi)$ stands for the distribution as estimated using ξ. As an example, in chapter 4 one of the aggregate statistics collected was DISTANCE FROM NEAREST ANT, which was calculated as the average distance to the nearest ant for all ants for all timesteps. This could be thought of as the first moment of the behavioral measure

$$m_{\overline{\text{NN}}}(\rho) = \frac{1}{N} \sum_i \|\mathbf{x}_i - NN(\mathbf{x}_i)\|$$

which computes the average distance between an agent and it's nearest neighbor over the entire group of agents at any specific time. This measure captures a notion of group cohesion, and while the first moment might summarize this aspect of behavior, its distribution can provide much more detailed information about the behavior: if $M_{\overline{\text{NN}}}$ is multi-modal, the agents might operate in multiple regimes, indicating a more complex behavior or a possibly

unforeseen change in or interaction with the environment.

While behavioral measures as described thus far encompass a broad class of functions, there are some measures of behavior that do not fit. For example, the comparisons in chapter 3 were made in terms of M_{xvel}, the distribution of forward velocity of each individual fish combined over the tracked period. That is, the histograms were computed by binning each agent's velocity at each timestep. As such, M_{xvel} does not fall within the class of functions described, but as chapter 3 showed, distributions like this can still be useful in comparing behavior models and comparing learned models. In order to distinguish between the two types of distributions in the remaining discussion, those that describe aspects of the entire group of agents will be called *group characteristics*, while those that describe individual agents will be referred to as *individual characteristics*. $M_{\overline{\text{NN}}}$ is an example of the first category, and M_{xvel} an example of the second.

With an appropriate collection of behavioral measures covering all the aspects of the behavior which are of interest, a quantitative measure of distributional similarity between two behaviors that can be computed from tracks, ξ and ξ^*, of agents performing these behaviors is given as

$$\zeta_{\xi^*}(\xi) = \sum_i D_{KL}(\hat{M}_i(\xi) \parallel \hat{M}_i(\xi^*)) \tag{5.2}$$

which in this thesis is termed Behavioral Divergence, and abbreviated BD. Here D_{KL} is the Kullback-Leibler divergence (Kullback and Leibler 1951), which is defined as

$$D_{KL}(P \parallel Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)} \tag{5.3}$$

for discrete distributions and

$$D_{KL}(P \parallel Q) = \int p(x) \log \frac{p(x)}{q(x)} dx \tag{5.4}$$

for continuous ones. Typically the estimation of $\hat{M}_i$ is accomplished by histogram, and so equation 5.3 can be directly applied.

Connections between Behavioral Divergence, Imitation Learning, and Behavioral Difference

Imitation Learning is another approach to learning models of behavior from samples of behavior. Historically, Imitation Learning is closely tied to the Reenforcement Learning community, and uses much of the same terminology, models, and techniques. A behavior of interest is modeled by a *policy* $\pi : \mathcal{S} \rightarrow \mathcal{A}$ which maps states to actions, or for non-deterministic behaviors, states to distributions over the action space $\mathcal{A}$. The goal of Imitation Learning is to learn a policy that produces trajectories that mimic some expert, given example trajectories drawn from the expert's policy. Recent work from the robotics community has demonstrated remarkable success by utilizing deep networks (Zhan et al. 2018), but has also illuminated just how challenging the problem can be.

Typically, an Imitation Learning approach attempts to find a policy which maximizes the expected likelihood, or equivalently minimizes the expected negative log likelihood:

$$\mathbb{E}_{\xi \sim \pi_e} \left[-\log p_{\pi_\theta}(\xi) \right] = \int_{\xi \in \Xi} -p_{\pi_e}(\xi) \log p_{\pi_\theta}(\xi) d\xi = H(P_{\pi_e}, P_{\pi_\theta}) \tag{5.5}$$

where $p_{\pi_e}(\xi)$ and $p_{\pi_\theta}(\xi)$ are the probability of a trajectory ξ under the expert and learned policy, and their distributions are denoted by P_{π_e} and P_{π_θ} respectively. Typically this objective is optimized with respect to some parameterization θ. This objective is also known as the *cross entropy*, and has a wide variety of applications across machine learning. The cross entropy is also related to the KL-divergence in the following way:

$$H(P, Q) = H(P) + D_{KL}(P \parallel Q)$$

where $H(P)$ is the entropy of distribution P. Since the expert's policy π_e is unaffected by changes to the parameters θ, this can be considered a constant offset, and so minimizing

the cross entropy is equivalent to minimizing the KL-divergence.

It is important to note that the domain of the integral in equation 5.5 is the space of *trajectories* – that is, a sequence of points in the state space – and so the optimization is based on matching the experts distribution of trajectories. This is typically justified with the assumption that actions are deterministic, but what is actually required is that a sequence of states unambiguously determine a sequence of actions. Either of these requirements are perfectly reasonable in certain domains, such as when trying to teach a robot to manipulate an object or complete a task, but they obfuscate the fact that there are no *sensors* in this model, which explicitly precludes any kind of collective behavior where an agents behavior relies on the sensed state of other agents. Some authors have addressed this issue either by operating in the feature space $\mathcal{O}$ instead of the state space Ξ, or by augmenting the model with some shared state for coordination (Zhan et al. 2018). In most cases, an analytical form of the expert policy distribution over trajectories is not known, although samples are readily available, and so the expectation in 5.5 is replaced by a finite sum over these samples.

Instead of using the sample trajectories to compute an estimate of the expectation, Behavioral Divergence can be viewed as estimating the divergence of the joint distribution of behavioral measures, *conditioned* on the observed tracks. Figure 5.1 illustrates this graphically. The assumption encoded by Figure 5.1 is that the distribution of each behavioral measure is conditionally independent of the other given the tracks that produced them. This is reasonable since, by construction, a behavioral measure must be computable from observed tracks. This independence assumption allows the KL-divergence of their joint distribution be decomposed into the sum of their individual KL-divergences

$$
\begin{aligned}
D_{KL}(M_1, M_2, \ldots, M_k \mid \xi \parallel M_1, M_2, \ldots, M_k \mid \xi^*) &= \sum_i D_{KL}(M_i \mid \xi \parallel M_i \mid \xi^*) \\
&\approx \sum_i D_{KL}(\hat{M}_i(\xi) \parallel \hat{M}_i(\xi^*)) = \zeta_{\xi^*}(\xi)
\end{aligned}
$$

Behavioral Divergence also has connections to earlier work focusing on quantitative measures of diversity in heterogeneous robot teams (T. Balch 2000), where the following is suggested as a quantitative measure of the dissimilarity of two agents:

$$D(f_a, f_b) = \int \frac{(p_a(o) + p_b(o))}{2} |f_a(o) - f_b(o)| do$$

T. Balch defines this as the *Behavioral Difference* between the agents a and b, where $p_a(o)$ is the probability of agent a encountering sensor values (or "perceptual state") o while acting according to its model (or policy) f_a. The definition is justified heuristically: the differences in behavioral response (output) are integrated over the space of possible sensor values (input) and weighed by the probability of encountering those sensor values. However, this can also be thought of as a special case of the first Wasserstein metric:

$$W(p_a, p_b) = \inf_{\gamma \in \Gamma(p_a, p_b)} \int_{\mathcal{O} \times \mathcal{O}} c(o, o') d\gamma(o, o')$$

Here, $\Gamma(p_a, p_b)$ represents the space of all joint probability distributions consistent with p_a and p_b being marginals of these joint distributions, and $c(o, o')$ is a cost function. If Γ is restricted to the singleton $\{\gamma_{1/2}\}$ where

$$\gamma_{1/2}(o, o') = \begin{cases} 0, & o \neq o' \\ \frac{p_a(o) + p_b(o')}{2} & \text{otherwise} \end{cases}$$

that is, a single joint distribution which assigns zero probability when $o \neq o'$, and equally mixes p_a and p_b when they are equal, and if the cost function is the absolute difference

$$c(o, o') = |f_a(o) - f_b(o')|$$

then it becomes clear that Behavioral Difference is a special form of the Wasserstein metric, and just like Behavioral Divergence it measures the distance between distributions,

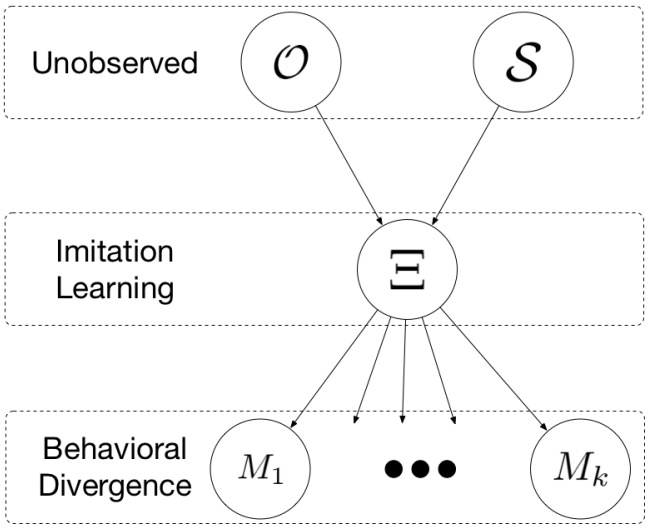

Figure 5.1: A probabilistic graphical model view of different approaches to minimizing divergence. The Imitation Learning objective (5.5) measures the divergence between the target (or expert) and the learned distribution of trajectories (the distribution of Ξ). Behavioral Divergence (5.2) measures the divergence between the target and learned distribution of behavioral measures, under the assumption that the distribution of these measures is independent given observed tracks ($\xi \in \Xi$). In both cases the true distributions of the observation model and the internal state are unobserved.

although since it is symmetric in its arguments it is a true distance and not a divergence.

Statistical divergences

The choice of KL-divergence for computing the similarity of the individual behavioral measures is motivated in the previous section through it's connection with existing learning objectives, but there are several other approaches to measuring the dissimilarity of two probability distributions that could be used in place of KL-divergence. The Wasserstein metric, mentioned in the previous section, is another example of measuring the dissimilarity of probability distributions. It is a generalization of the "earth-movers distance" (EMD) to continuous probability distributions: a way of describing the minimum amount of work to transform one distribution into another, which has seen wide success in the computer vision community (Rubner, Tomasi, and Guibas 2000). Recent work on Generative Adversarial Networks (GANs) has used the Wasserstein metric as a learning objective, showing success

in improving some aspects of the learned models (Arjovsky, Chintala, and Bottou 2017). This is attributed to the convergence properties of the Wasserstein metric, as compared with alternatives (including KL-divergence). The trade off for these improved convergence properties is the intractability of computing the infimum. Computing this infimum can be avoided by using a differentiable upper bound, which is useful for a learning objective, but less interpretable as a stand alone measure of similarity.

The *Total Variation* (TV) distance, defined as

$$\delta(P, Q) = \sup_{A \in \Sigma} |P(A) - Q(A)|$$

where the supremum is taken over all possible events for both distributions with σ-algebra Σ. Rather than quantify the similarity of the two distributions in terms of the minimum cost of transforming from one to the other, δ captures the maximum discrepancy. TV and KL-divergence are related by Pinsker's inequality: $\delta(P, Q) \leq \sqrt{\frac{1}{2} D_{KL}(P \parallel Q)}$. Arjovsky, Chintala, and Bottou show that the learning method presented in the well known energy-based GAN (Zhao, Mathieu, and LeCun 2016) is equivalent to optimizing a TV learning objective.

Many other methods for comparing histograms, such as Chi-squared distance or Jaccard distance[1], have been proposed and used in a variety of specific domains (Cha 2007), and although few are as well studied and have as direct a relationship to fundamental properties of probability distributions as KL-divergence, this work does not intend to claim that one measure of statistical similarity is superior in all cases. The claim is merely that KL-divergence is sufficient as a measure of similarity between distributions for the purposes of quantifying the quality of a learned model of behavior in terms of distributional similarity.

[1] also sometimes referred to as Intersection-over-Union (IoU)

Behavioral measures

The individual components of Behavioral Divergence, called behavioral measures and denoted by $\hat{M}_i$ are the core of what Behavioral Divergence is actually measuring. Beyond the requirements described in section 5.1, such as being computable from tracks, little has been said about how to select these measures for a given task. In previous chapters, examples of behavioral measures have been given that are intuitively related to aspects of group behavior that seem to be important in a qualitative sense. The following will describe those behavioral measures and their connection to aspects of group behavior explicitly:

- **Group diameter**: computed as the maximum distance between any two members in the group, this captures the overall group size.

- **Group density**: computed as the average distance to the nearest other member of the group, this captures how closely packed the group is.

- **Group uniformity**: computed as the variance in the distance to the nearest other group member, this captures the regularity of the distribution of the group members within the group.

Certain aspects of the distribution of each of these measures can describe aspects of the group behavior. For example, distributions that are tightly clustered about multiple modes indicate that the group might be alternating between multiple low level behaviors, or that the collective behavior goes through multiple phases.

Other behaviors in other domains may suggest alternative measures, some of which will be described in later chapters. In general some domain knowledge is helpful to identify appropriate behavioral measures in practice.

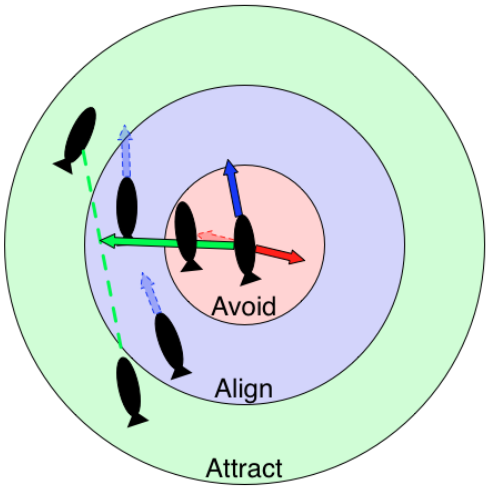

Figure 5.2: An illustration of the sensor model described in Couzin et al. 2002. Each concentric zone effects a different aspect of the agents motion: separation, alignment, and cohesion (nearest to furthest).

Assessing schooling behaviors in fish

The remainder of this chapter will be focused on presenting how Behavioral Divergence can be implemented and used in a simple domain: assessing the quality of learned models of fish schooling behavior. In the first experiment, a synthetic schooling behavior is used to generate tracks from which a successful executable model is learned. In the second, live tracking data of real fish is used to train a model which produces behavior that is not a good match qualitatively. Behavioral Divergence captures the quality of the match in both cases.

Experiments

Both experiments follow the same outline:

1. Obtain tracks of the animals performing schooling behavior
2. Learn an executable model given the tracking data using the k-NN method described in chapter 3.
3. Run the learned model in simulation, collecting tracks.

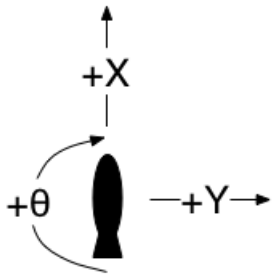

Figure 5.3: The fish actuator model. Fish can move forward/backward, laterally, and turn in place. The motion of the fish can be computed by comparing consecutive points in the tracking data.

4. Compare the learned behavior using three behavioral measures (described below), and from this compute the Behavioral Divergence between the learned behavior and the generating behavior.

In the first experiment the tracking data is generated from simulation logs, and in the second experiment it is computed from video of live animals via a multi-target tracking algorithm (Sbalzarini and Koumoutsakos 2005). The same sensor (Figure 5.2) and dynamics (Figure 5.3) models are used in both experiments.

Schooling metrics

As discussed earlier, the behavioral measures used to evaluate the behaviors should be computable from observation data, and should cover some import aspects of the behavior of interest. For fish schooling, three measures which describe aspects of the entire school are:

1. **Maximum distance** between two fish in the school. This is an indicator of the overall school diameter

2. **Average distance** to the nearest neighbor fish. This describes the density of the school.

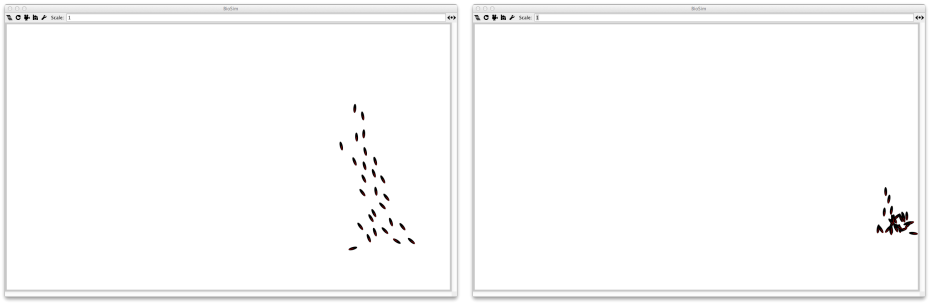

Figure 5.4: Screenshots of the simulator showing the training behavior (*left*) and the learned behavior (*right*)

3. **Variance** in the average distance above. This describes how uniformly the school is distributed.

Each of these can be computed efficiently from tracking logs, as can histogram estimates of their distributions. These can then be plugged in to equation 5.2 to compute the Behavioral Divergence between the learned models and the behaviors which generated the training data, quantitatively characterizing the similarity of the compared behaviors.

Synthetic schooling

The purpose of the first experiment is to show that the Behavioral Divergence of two similar behaviors is small, indicating that the behaviors are indeed similar, and to quantitatively characterize the range of values to expect for similar behaviors. To do this, a simple synthetic schooling model was developed to provide an easily learnable target behavior for which ground truth was available.

The schooling behavior described in Couzin et al. 2002 is one of many conceptually similar models of "flocking" behavior famously described in Reynolds 1987 where the motion of individual agents is affected by nearby flock-mates in several different ways. The three components of this model are

- **Separation**: Agents prefer to be separated from each other by a minimum distance.
- **Alignment**: Agents prefer to head in the same direction as the rest of the school.

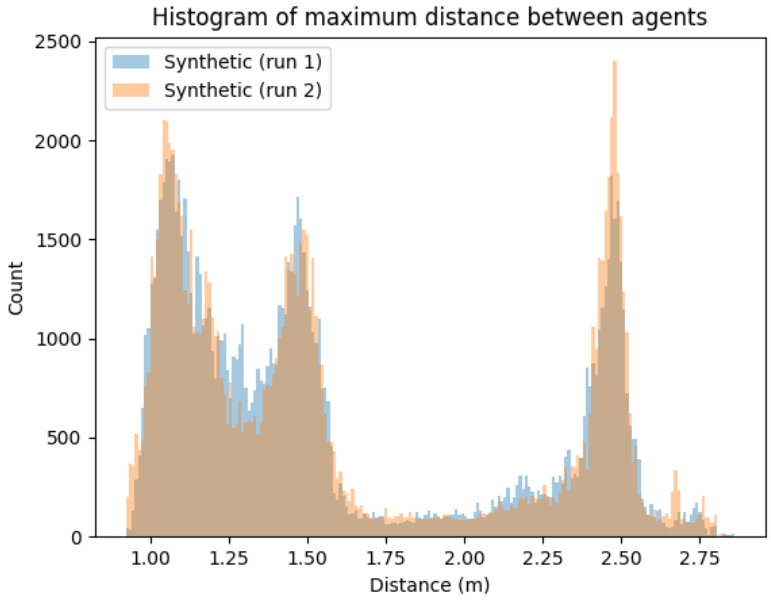

Figure 5.5: The distribution of maximum distance for the same synthetic behavior from two separate runs. The figures show that the distributions are almost identical, as is expected, and that variation in starting conditions which might lead to drastically different final positions does not significantly impact the distribution of the behavioral measures.

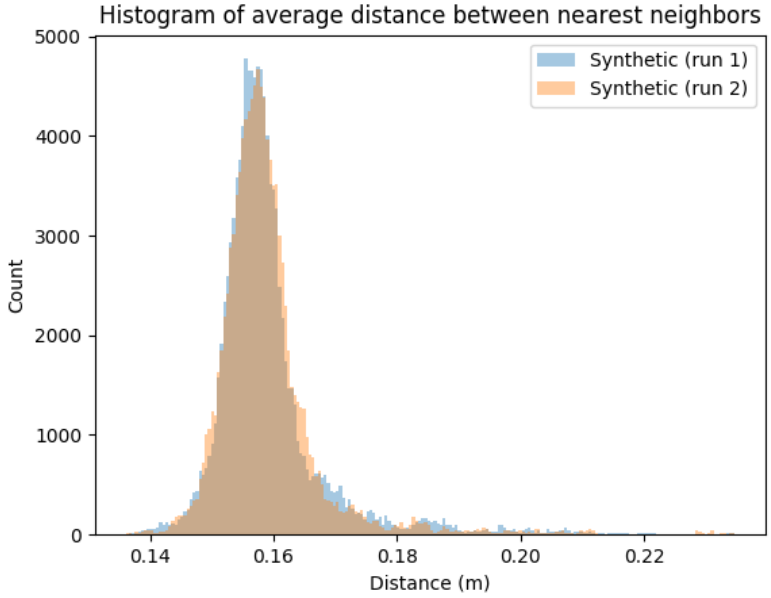

Figure 5.6: The distribution of average nearest neighbor distance for the same synthetic behavior from two separate runs. The figures show that the distributions are almost identical, as is expected, and that variation in starting conditions which might lead to drastically different final positions does not significantly impact the distribution of the behavioral measures.

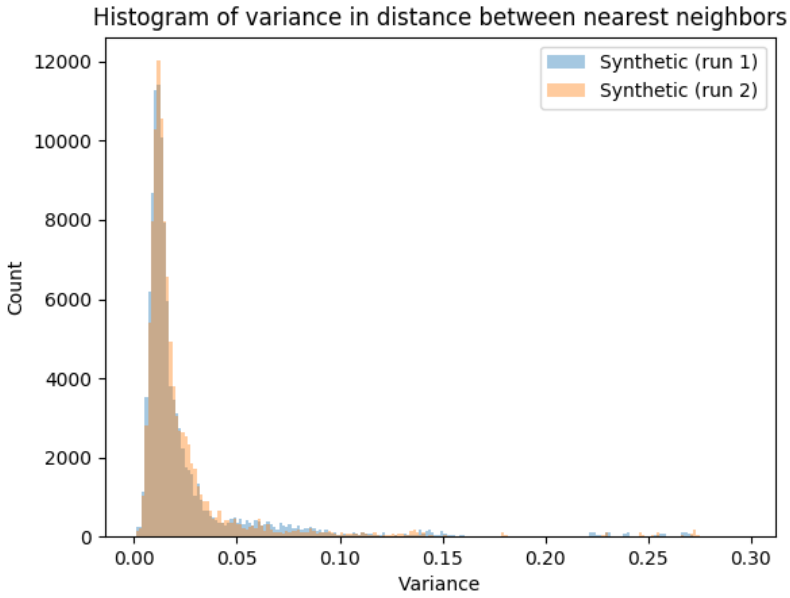

Figure 5.7: The distribution of the variance in nearest neighbor distance for the same synthetic behavior from two separate runs. The figures show that the distributions are almost identical, as is expected, and that variation in starting conditions which might lead to drastically different final positions does not significantly impact the distribution of the behavioral measures.

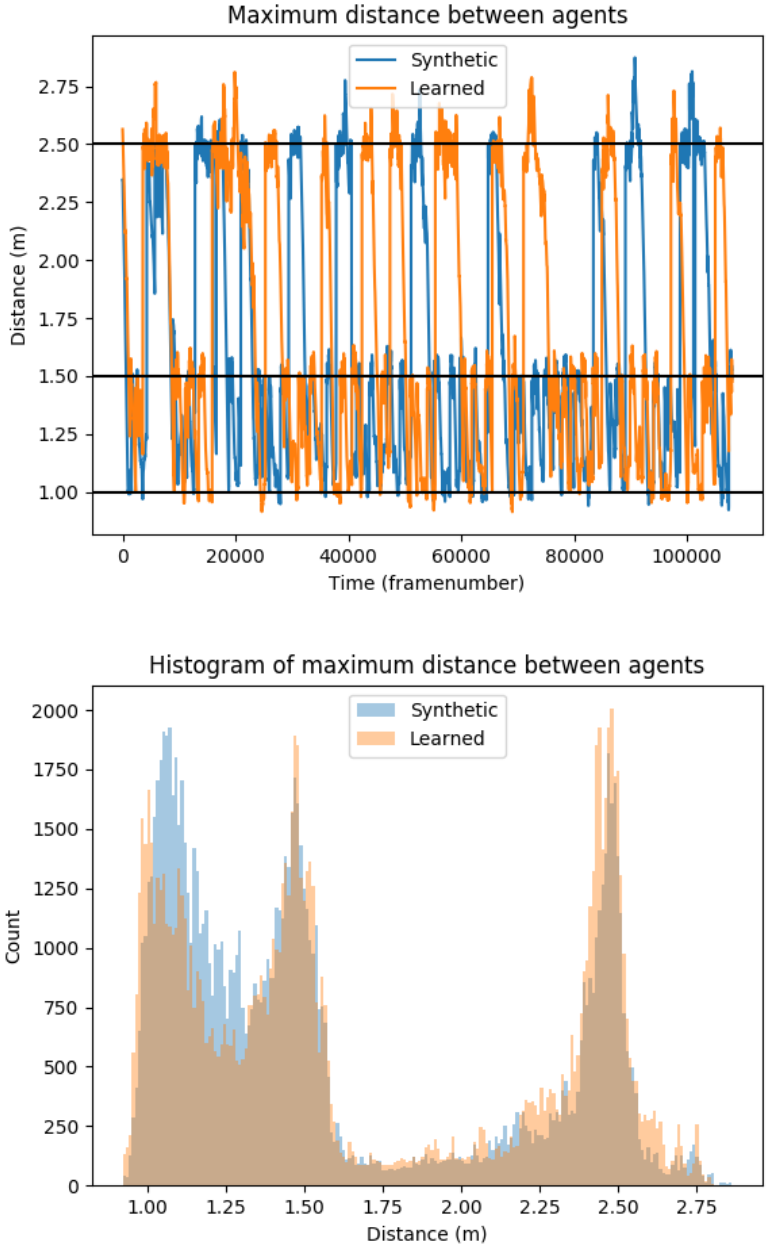

Figure 5.8: Synthetic (blue) and learned (orange) behavior. Timeseries (top) and histogram (bottom) of the maximum distance between any two agents, a measure of the diameter of the group. The horizontal black lines in the timeseries plot correspond with the modes of the histogram.

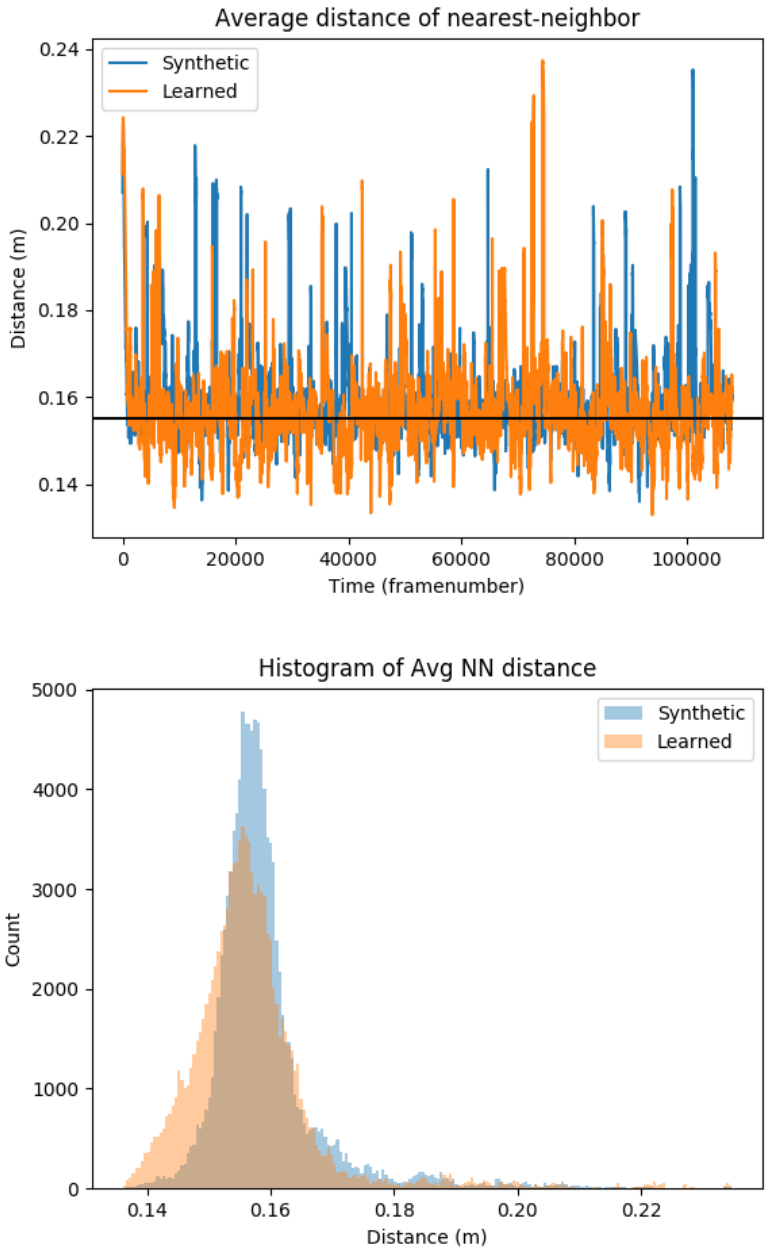

Figure 5.9: Synthetic (blue) and learned (orange) behavior. Timeseries (top) and histogram (bottom) of the distance between each agent and its nearest neighbor averaged across all agents, a measure of the density of the group. The horizontal black line in the timeseries plot corresponds with the mode of the histogram.

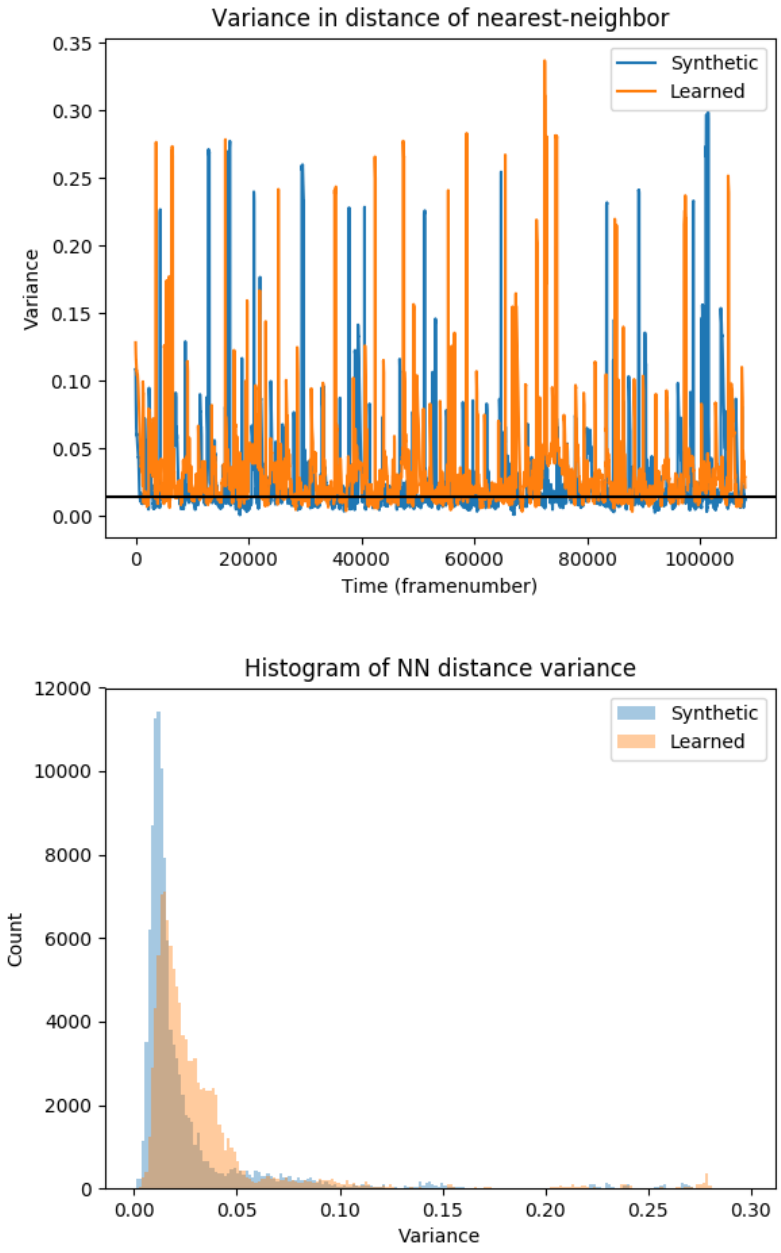

Figure 5.10: Synthetic (blue) and learned (orange) behavior. Timeseries (top) and histogram (bottom) of the variance in the distance between agents and their nearest neighbor, a measure of the uniformity of the group. The horizontal black line in the timeseries plot corresponds with the mode of the histogram.

- **Cohesion**: Agents prefer to be near the center of the school.

An agent computes the separation component by negating the vector towards the center of mass of flock mates within a distance threshold. The alignment component is computed as the average normalized velocity of flock mates past the separation distance threshold, but within an alignment distance threshold. Finally, the cohesion component is a vector towards the center of mass of all the flock mates within sensor range, but beyond the alignment distance threshold. Figure 5.2 illustrates graphically these three components. From these components, an agent computes an overall desired heading which it steers towards while maintaining a constant forward velocity.

The simulation of schooling fish was developed using the behavior described above. Figure 5.4 is a screenshot of the simulation which was built using BioSim[2]. The simulation consisted of 30 fish in a toroidal environment, and ran for 60 simulated minutes. Data was collected at each time step (simulated 30Hz) of the position and orientation of each fish, to simulate the output of a multi-target tracking algorithm. From this data, the range and bearing was computed for each of the three vectors (separation, alignment, and cohesion), and combined them into a 6-component feature vector for each fish at each time step. These feature vectors were then matched with the change in position and orientation to create sequences of about 2.6 million input/output pairs. These sequences were then used to train an executable model from the data using the kNN-Sample approach described in chapter 3.

Figures 5.8, 5.9, and 5.10 show the evolution of the three schooling metrics at each time step, and give the histograms for them illustrating that the behavior of the learned model closely matches the synthetic behavior. Table 5.2 lists the KL-Divergence of each of the three histograms.

To provide context for the numerical values presented, Table 5.1 and Figures 5.5, 5.6, and 5.7 give the same results for two runs of the synthetic behavior. Both runs started with

[2] `https://github.com/biotracking/biosim2`

Table 5.1: KL-divergence between two runs of the same synthetic behavior. This serves to validate that BD is low when the behaviors are generated from the same underlying model, and to give a quantitative reference for the values of BD to expect when the behaviors are very similar. The Behavioral Divergence value in this case is 0.266.

Max distance	Avg NN distance	Var NN distance
$D_{KL} = 0.043$	$D_{KL} = 0.091$	$D_{KL} = 0.132$

Table 5.2: KL-divergence for synthetic schooling. Behavioral Divergence in this case is computed by calculating the sum of the KL-Divergences for the three behavioral measures: school diameter, school density, and school uniformity. These three behavioral measures are estimated by computing histograms of the maximum distance between agents in the school, the average distance between nearest neighbors, and the variance in nearest neighbor distance. The Behavioral Divergence value in this case is 0.731, close to the value reported in 5.1 for identical behaviors. This indicates that the learned model is reproducing the same behavior as is demonstrated in the training data.

Max distance	Avg NN distance	Var NN distance
$D_{KL} = 0.074$	$D_{KL} = 0.138$	$D_{KL} = 0.519$

different initial conditions, but used the same underlying behavior.

Real schooling

The purpose of the second experiment is to show how Behavioral Divergence quantifies the dissimilarity of two behaviors that are qualitatively not similar: the behavior of real fish and the behavior of a trained executable model. The training data in this experiment were taken from tracks[3] of 30 juvenile *Notemigonus Crysoleucas* or "Golden Shiners" schooling in a shallow tank 2.1 meters long by 1.2 meters wide. Figure 5.11 shows the fish in this environment. The same features were used as described in the previous experiment with the addition of the range and bearing to the nearest obstacle to account for the addition of walls to the environment. The same process for compiling input/output pairs to train the executable model was used, and the same behavioral measures were computed. Figures

[3]This data was taken from one replicant of the experiments performed in Katz et al. 2011.

Figure 5.11: A frame from the video of the fish in their shallow tank. Image kindly provided by Iain Couzin and the Collective Animal Behavior Lab at Princeton University.

5.12, 5.13, and 5.14 show the timeseries and histograms describing the distribution of the three metrics, but unlike the synthetic case, the two behavior models are not similar, as shown quantitatively in Table 5.3.

Qualitatively this dissimilarity manifests in several ways. The trained model produces a behavior where the individuals can get stuck on the walls of the environment, leaving them far away from the school. This can be seen in the significant rightward shift of the maximum distance behavioral measure. One reason that the learned model does not avoid walls more aggressively is that in the training data, the fish never actually get close enough to the walls to collide. Since the fish do not ever visit regions of the feature space close to the walls, there is little data in these regions to guide the learned model, an issue common in the Imitation Learning community which has developed some methods to address it (Venkatraman, Hebert, and J. A. Bagnell 2015; Ross, G. Gordon, and D. Bagnell 2011). Chapter 6 will return to explore these techniques, and how they interact with Behavioral Divergence.

Another possible reason for the mismatch in the behavior of the trained model with the behavior of real fish is that the selected features ϕ do not capture all the important

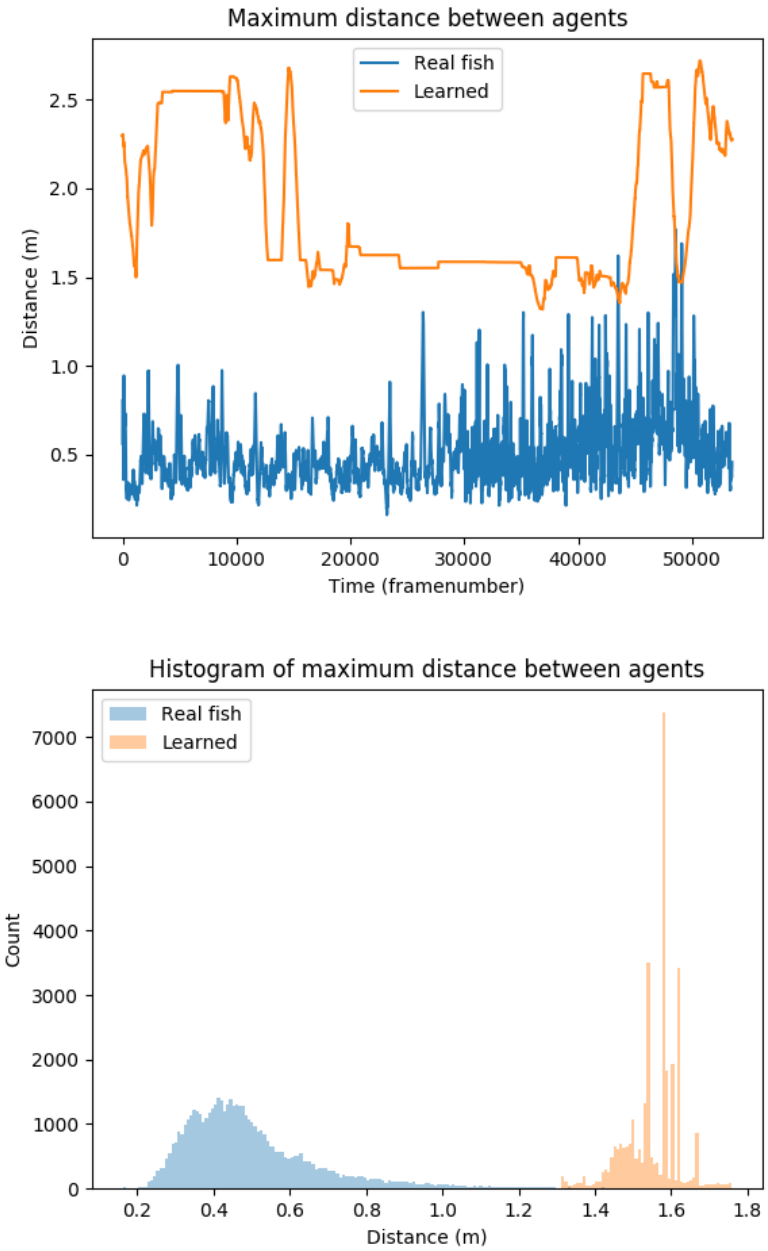

Figure 5.12: Real (blue) and learned (orange) behavior. Timeseries (top) and histogram (bottom) of the maximum distance between any two agents. Unlike the synthetic case, the learned behavior does not match the real behavior.

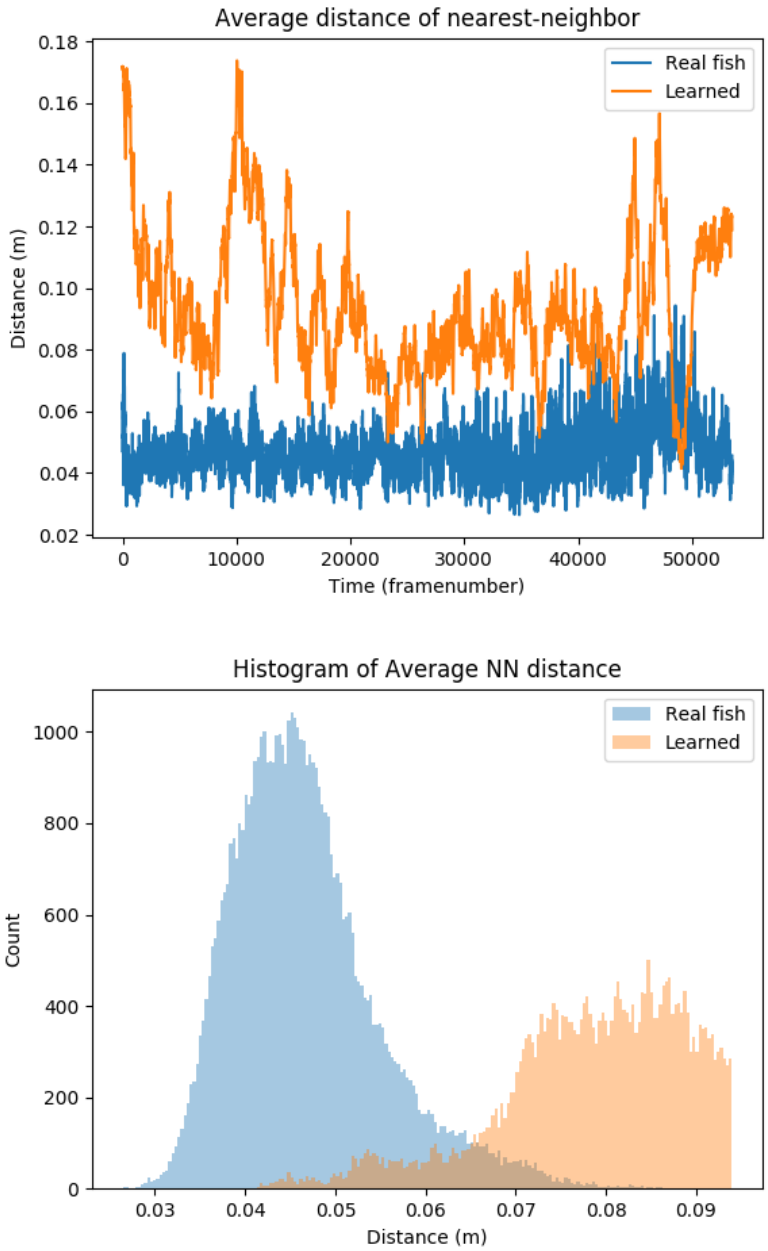

Figure 5.13: Real (blue) and learned (orange) behavior. Timeseries (top) and histogram (bottom) of the average nearest-neighbor distance. Unlike the synthetic case, the learned behavior does not match the real behavior.

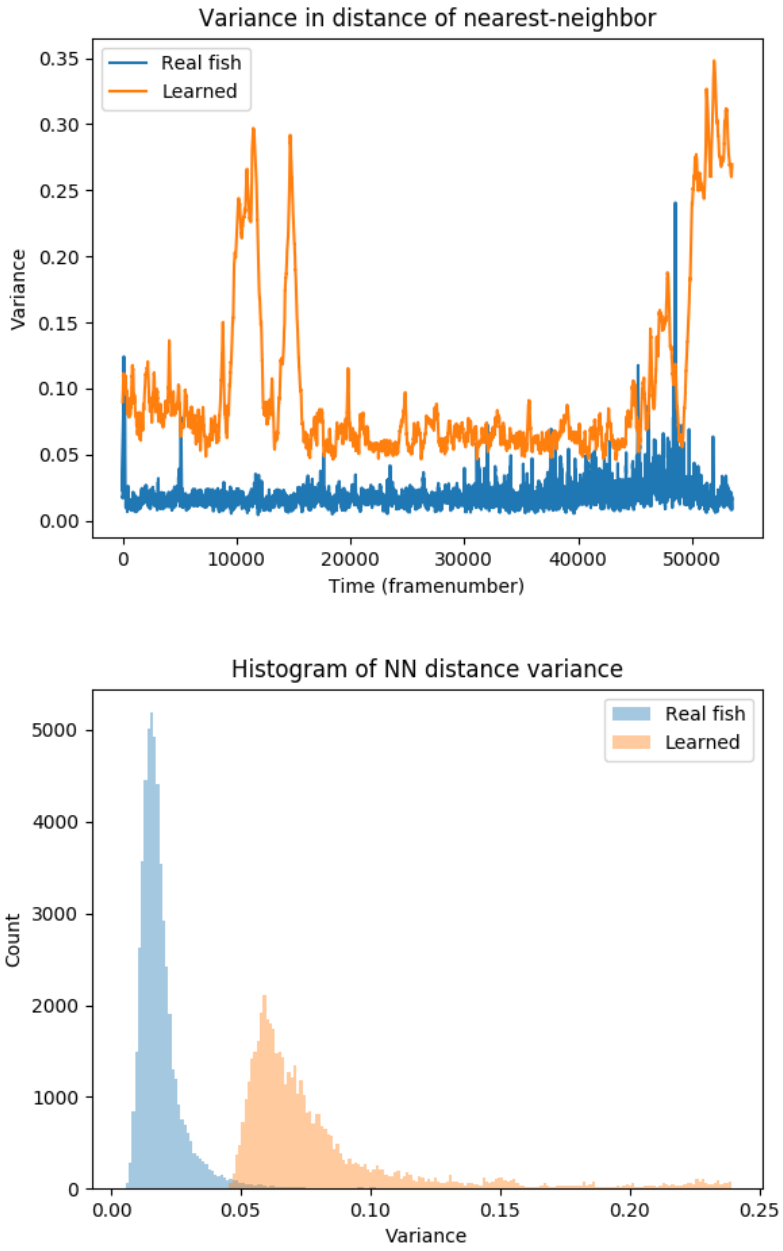

Figure 5.14: Real (blue) and learned (orange) behavior. Timeseries (top) and histogram (bottom) of the variance of the nearest-neighbor distance. Unlike the synthetic case, the learned behavior does not match the real behavior.

Table 5.3: KL-divergence for real schooling. Behavioral Divergence in this case is computed by calculating the sum of the KL-Divergences for the three behavioral measures: school diameter, school density, and school uniformity. These three behavioral measures are estimated by computing histograms of the maximum distance between agents in the school, the average distance between nearest neighbors, and the variance in nearest neighbor distance. The Behavioral Divergence value in this case is 73.99, considerably higher than in the synthetic case, indicating that the learned model is not reproducing the same behavior as is demonstrated in the training data.

Max distance	Avg NN distance	Var NN distance
$D_{KL} = 31.438$	$D_{KL} = 10.293$	$D_{KL} = 32.259$

aspects of the environment that influence the real behavior of the fish. Another qualitative difference between the two behaviors that was observed was the tendency of the real fish to alternate between schooling in one region of the environment and moving as a group to another region. This bi-modal behavior might be explained by a feature the real fish are sensitive to that the learned model does not compute, or possibly some internal behavioral state which would be better handled by a BIOHMM model as described in chapter 4.

Using Behavioral Divergence with more complex behaviors

While for this chapter it is sufficient to note that Behavioral Divergence can distinguish between similar and dissimilar behaviors, it would be beneficial to examine how BD behaves in more detail. It would be particularly useful to understand the relationship between BD and measures of predictive performance. The next chapter presents a more detailed experimental analysis of the behavior of BD within the context of learning a model of a more complicated behavior, namely playing team soccer.

CHAPTER 6

APPLYING BEHAVIORAL DIVERGENCE: A CASE STUDY IN SOCCER

Chapter 5 formally defined Behavioral Divergence and provided a brief example of its use in evaluating the relative quality of different executable models of behavior in the case of schooling fish. This chapter focuses on a more in-depth and complex example: learning an executable model of soccer playing behavior.

Additionally, while the goal in previous chapters has primarily been on learning an accurate model from tracks, this chapter is more concerned with the properties of BD than the specifics of how the model is learned. For this reason the coming chapter will make use of more powerful learning tools that would not be applicable in all the situations presented previously. Specifically, in the context of Learning from Demonstration it is frequently assumed that the learner has the ability to query "expert examples" to varying degrees. Access to an expert can substantially improve performance but can be an unrealistic assumption, particularly in cases of animal behavior. In this instance, since the goal is to assess the properties of Behavioral Divergence rather than the model, this is a reasonable approach.

Case study: Robot soccer

Robot soccer presents an interesting domain for learning executable models of behavior. Team behaviors can exhibit a wide range of complexity, while metrics of success, such as points scored, are easy to compute and straightforward to interpret. By designing a reasonably complex team behavior, one that is feasible but not trivial to learn, it is possible to compare how Behavioral Divergence tracks with successful learning (as determined by qualitative examination and quantitative external metrics, such as goals scored) and measures of predictive performance (like RMSE).

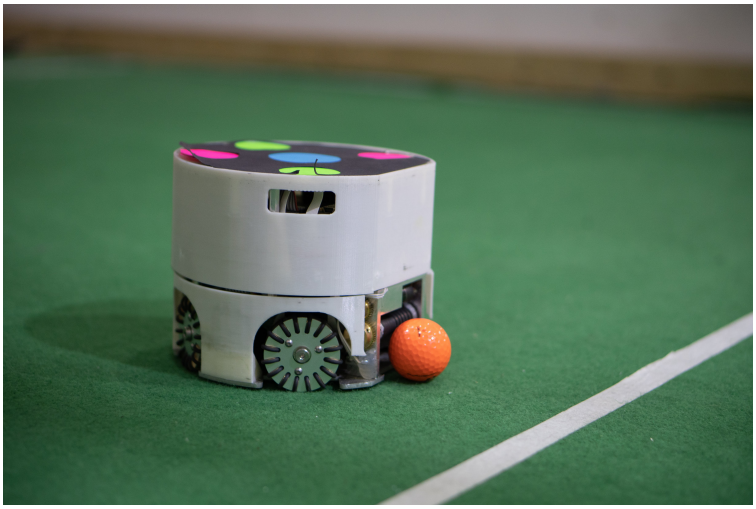

Figure 6.1: A robot built for the Robocup Small-Size League competition by the Georgia Tech RoboJackets organization.

Feature design for soccer behavior

The features available to the individual agents play an important role in how the agents can react to their environment. For the task of team soccer play, the following perceptual features were selected to allow for a simple implementation of a straightforward behavior that still exhibits some complexity:

- `ball`: a vector to the ball.
- `oppgoal`: a vector towards the center of the opposing team's goal
- `teamgoal`: a vector towards the center of the agent's goal
- `nearestopp`: a vector towards the nearest opponent
- `nearestteam`: a vector towards the nearest teammate
- `gripperp`: a boolean indicator for if the agent has grasped the ball
- `closestp`: a boolean indicator for if the agent is the closest to the ball on it's team

- `tgclosestp`: a boolean indicator for if the agent is the closest to the team goal on it's team
- `aimedatgoalp`: a boolean indicator for if the agent is oriented towards the opponent's goal
- `aimedatteamp`: a boolean indicator for if the agent is oriented towards it's nearest other teammate

For all vector features, the vector is represented as the ego-centric x and y coordinates. For all boolean features, true and false are represented by $+1.0$ and -1.0 respectively. The `aimedatgoalp` is true if the ray originating at the center of the agent and extending along it's ego-centric x-axis intersects with a circle centered at the opponents goal with diameter equal to the width of the goal, and false otherwise. Similarly, `aimedatteamp` is true if the same ray intersects with a circle centered at the nearest teammate with diameter equal to the diameter of the agent.

Learning from tracks of real robot soccer

Robocup[1] is an annual international competition between robot soccer teams modeled after the world cup. There are a variety of "leagues" for different form factors and levels of autonomy, and a growing number of leagues addressing non-soccer specific challenges. Teams participating in the Small Size league build their own platforms (an example is given in Figure 6.1), but use a common perception system which is maintained and operated by the competition organizers, and logs from all competition games are available.

Unfortunately, learning models of the Robocup robots from these logs is not so straightforward. First, the logs themselves contain significant noise. While for the most part noise in the position of each agent can be smoothed away, orientation noise is significant. An ambiguous orientation is problematic because the orientation determines the direction of

[1] `http://www.robocup.org/`

Table 6.1: Performance of models learned from Robocup data. Behavioral Divergence is computed using tracks from the logs of Robocup data, tracks of the learned model, and a withheld set of Robocup tracks of the same behavior. The Robocup logs are much more similar to the withheld test set than the tracks generated by the learned behavior, indicating that the learned model is not accurately reproducing the behavior it was trained from.

	Withheld Data	Learned Model
Behavioral Divergence	2.287	20.643

any kicks. Additionally, while the robots themselves are large enough so that fluctuations in position are small relative to their physical extent, the ball is much smaller, and exhibits significant tracking error, such that smoothing makes the detection of collisions, kicks, and passes extremely difficult.

A second difficulty lies in the kind of play adopted by most teams in the competition. The most successful teams make significant use of "set plays", where the team in control of the ball plans out a set of trajectories, passes, and kicks, designed to optimize their likelihood of scoring a goal. This type of behavior involves a considerable amount of planning. While recent work in "learning to plan" such as Tamar et al. (2016) has made progress on learning systems which develop planning submodules, the task in general is quite difficult, and in general requires access to substantially more data than is available in this domain.

In preliminary testing it was clear that the models learned from this data were not able to reproduce the generating behavior. An early test which used one game's worth of training data (roughly 200,000 samples) resulted in behavior with significant Behavioral Divergence, as shown in Table 6.1, as well as obvious qualitative failures such as running into walls, not approaching/kicking the ball, and attempting to kick without the ball. Interestingly, the 10 fold cross-validation RMSE was relatively low with a mean of 0.051 and standard deviation of 0.007, which is in the same range as later results. The implication that Behavioral Divergence is more sensitive, or at least sensitive to different features of the output of the learned model will be explored more fully in subsequent experiments.

For these reasons, the remainder of this chapter will focus on analyzing Behavioral

Divergence using a simple synthetic behavior described in the next section.

Target soccer behavior design

Team soccer serves as the domain for this set of experimental investigations into Behavioral Divergence. Simulations of two teams of synthetic agents playing head to head in simulation provide the training data. The behavior of these hand-coded agents described below is the target behavior to be learned.

The target synthetic behavior is intended to be more complex than the simple fish schooling behavior described in the previous chapter, while still providing an achievable learning objective. To avoid issues with local optima and computational complexity common in learning models with behavioral state, the behavior is designed to be entirely reactive, making no recourse to internal state, although it will be convenient to logically think of the behavior consisting of three sub-behaviors:

- STRIKER: The agent moves towards the ball and attempts to kick it towards the opponents goal or an open teammate.
- GOALIE: Follows the same behavior as STRIKER, but stays within two meters of the team goal.
- RECEIVER: Orients towards the ball moving slowly towards it while avoiding other teammates and opponents.

Which of these three sub-behaviors is active depends only on the boolean features, so there is no need for internal or behavioral state. If `closestp` is true, the agent follows the STRIKER behavior. Otherwise, if `tgclosestp` is true, the agent follows the GOALIE behavior. If neither is true, the agent follows the RECEIVER behavior. Once the STRIKER has gripped the ball, it stops lateral motion and begins rotating in place to orient towards either an open teammate or the opponent's goal. If the nearest teammate is closer to the opponents goal and is not blocked by the nearest opponent, determined by checking if the

nearest opponent is close enough to the line segment connecting the agent and the nearest teammate to collide with the ball, the agent attempts to kick the ball in the direction of the nearest teammate. Otherwise, the STRIKER kicks the ball in the direction of the opponents goal.

Learning methodology for soccer behavior with expert demonstrations

Since the synthetic behavior generating the tracks does not rely on behavioral state, a single model can be learned using the methods described in chapter 3, specifically kNN-Sample. However, unlike the in case of fish, there are significantly more features and their geometric relationship is more complex. The boolean features are essentially discrete and the vectors based on nearest teammate or nearest opponent can quickly change in response to very small motion, both of which cause sharp boundaries. While this might be addressed through feature engineering, the purpose of this chapter is to explore the properties of Behavioral Divergence, rather than solving the specific problem of learning team soccer. Additionally, using tracks of sample behavior results in a very sparsely sampled training set. In addition, the error in the learned model's output can feed back in to the system by moving the agents to regions in the feature space that are far from any training points.

This is a fairly common problem in Imitation Learning. Ross, G. Gordon, and D. Bagnell (2011) present a method they call "Dataset Aggregation" (DAGGER) which, given access to an expert f^* that can be queried at arbitrary points in the feature space, produces a sequence of models $\{f_i\}_{i=1}^N$. For each iteration, a model trained on the current training dataset is then run in simulation for a fixed number of steps T, generating a sequence of environmental states. The expert's action is then queried for each point in the trajectory, producing a new set of training tuples which are added to the training data for the next iteration. Algorithm 1, reproduced from Algorithm 3.1 in Ross, G. Gordon, and D. Bagnell (2011), describes this with a few more details. First, the models are assumed to be drawn from some class $\mathcal{F}$ determined by the representation and underlying learning method. Sec-

ond, the line "Let $f_i = \beta_i f^* + (1-\beta_i)\hat{f}_i$" allows for blending between the expert and current trained model to give more control over what model is used to generate the new trajectory at iteration i. One common way to blend these models mentioned by Ross, G. Gordon, and D. Bagnell is to set $\beta_1 = 1$ and $\beta_t = 0, \forall t > 1$. This sets the first training dataset to be completely from the expert, and the remainder completely from the most recent trained model. This will be the method used in the experiments below.

Algorithm 1: the DAGGER algorithm. Reproduced from Algorithm 3.1 in Ross, G. Gordon, and D. Bagnell (2011) with slight adjustments for terminology.

Initialize $\mathcal{D} \leftarrow \emptyset$.
Initialize $\hat{f}_1$ to any model in $\mathcal{F}$.
for $i = 1$ *to* N **do**
  Let $f_i = \beta_i f^* + (1 - \beta_i)\hat{f}_i$.
  Sample T-step trajectories using f_i.
  Get dataset $\mathcal{D}_i = \{(e, f^*(\phi(e)))\}$ of states visited by f_i and actions given by expert f^*.
  Aggregate datasets: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$.
  Train model $\hat{f}_{i+1}$ on $\mathcal{D}$.
end
return best $\hat{f}_i$ on validation.

Experiments in assessing learned synthetic soccer behavior

In order to empirically examine the behavior of Behavioral Divergence and compare it with predictive performance, the following experiments were performed. The simulation engine and environment used in these experiments was written from scratch using BioSim[2]. Tracks of the behavior described in section 6.4 were collected by running a simulation of ten agents (five per team) in a 9m long by 5m wide arena surrounded by walls. Goals were placed at opposite ends along the length of the arena, 10cm from the near wall, 18cm deep and 1.2m wide, centered along the width of the arena. A 4cm diameter soccer ball was placed in a random position within the arena at the start of the simulation, and reset randomly for every goal. Simple elastic collisions between the ball and robots or walls

[2] `https://github.com/biotracking/biosim2`

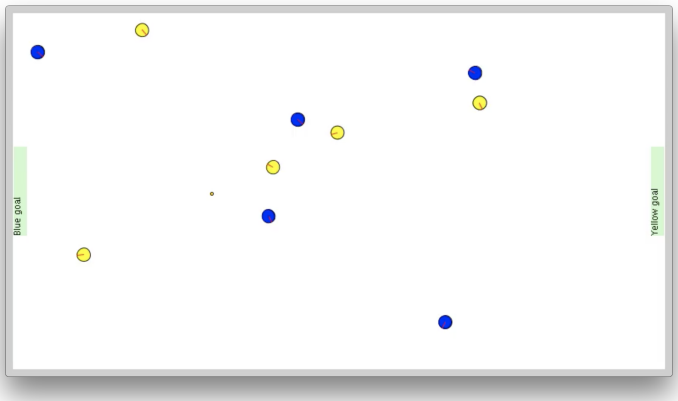

Figure 6.2: A screenshot of the simulation environment for the robot soccer experiments.

was modeled, while robot-robot and robot-wall collisions simply prevented translational robot motion. The robots, each 18cm in diameter, were initially randomly placed within the arena. The simulation was run at a temporal resolution of 30 steps per second. The simulation runs were broken into "games" with each consisting of 10 simulated minutes of play, which corresponds to 180,000 feature/output pairs. Figure 6.2 shows a screenshot from the simulation.

Using this simulation and training data collection setup, models were trained with varying amounts of data, from 180,000 to 3,780,000 samples using kNN-Sample. Models were also trained using DAGGER, with kNN-Sample as the "inner" learner, starting from 180,000 samples generated from simulation, and then collecting 9,000 expert samples per DAGGER iteration for 1 to 400 iterations. Once these models were trained they were then run in simulation against the ground truth behavior in 10 games, and statistics on the goals scored per 10 minute game, as well as the following behavioral measures were collected for computing Behavioral Divergence:

1. Average distance to team goal. This is useful for capturing the non-GOALIE behaviors, as they should be distributed fairly evenly around the ball. This also can provide proxy information on the location of the ball.

2. Minimum distance to team goal. This is useful for capturing the GOALIE behavior, since the GOALIE is determined by being the closest to the team goal.

3. Average distance to nearest teammate. This is useful for capturing the RECEIVER behavior, as the RECEIVERs should be spreading out from nearby teammates.

4. Average distance to nearest opponent. This is useful for capturing the RECEIVER behavior, as the RECEIVERs should be spreading out from nearby opponents.

5. Minimum distance to nearest opponent. This is useful for capturing the STRIKER behavior, because both team's STRIKER should be headed for the ball, which puts them within close proximity to each other frequently.

6. Average distance to ball. This is useful for capturing the RECEIVER behavior, because most of the team should be spreading out around the ball.

7. Minimum distance to ball. This is useful for capturing the kicking aspects of the STRIKER behavior, since the STRIKER should be the closest member of the team to the ball and should be quickly kicking it towards the opponents goal or an open teammate.

In all of the experiments reported below, each of these measures is estimated from tracks of the learned and target behaviors with histograms. In each case where Behavioral Divergence is reported the value is computing as the summed KL-divergence for each measure.

Finally, one game was logged to use as a withheld test set for comparison purposes.

Comparing Behavioral Divergence and predictive performance

The experiments discussed in this section are intended to answer the following five questions:

1. Given an initial model, does performing DAGGER iterations qualitatively improve the behavior of the learned model? It is expected that DAGGER does improve performance qualitatively because the expert samples it generates taken from the distribution of feature values the learned model is likely to see, thus "correcting" incorrect actions.

2. Does Behavioral Divergence of the learned model improve more with subsequent iterations of DAGGER compared to an equivalent number of plain training examples? It is expected that Behavioral Divergence will improve more with DAGGER than plain training examples because Behavioral Divergence reflects or correlates with the qualitative improvements seen in the previous experiment.

3. Does the per-sample RMSE of the learned model on samples taken from the same distribution as the training data improve more with DAGGER than an equivalent number of plain training samples? Again, it is expected that RMSE will improve more with DAGGER than plain training examples because the DAGGER samples are correcting mistakes that the learner is likely to make.

4. Does the per-sample RMSE on a withheld test set taken from the distribution of features generated by the target behavior improve with increased training samples? Since the samples generated by DAGGER do *not* come from the same distribution as the target behavior, it is expected that performance on a withheld test set will *not* significantly improve when adding training data, either from DAGGER or plain.

5. How will the distribution of each of the behavioral measures which makes up the computed value for Behavioral Divergence change with subsequent iterations of DAGGER? Since DAGGER is not directly optimizing Behavioral Divergence or any of the component behavioral measures, it is expected that the learned behavior will be more similar according to some measures, while others may become less similar or shift without significantly changing their KL-divergence.

The initial behavior learned on 180,000 samples of the synthetic soccer behavior is able to reproduce several aspects of the behavior, but fails to accurately reproduce the entire behavior, as is illustrated in the top graphs of Figures 6.3, 6.7, 6.8, 6.9, 6.10, 6.11, 6.12, and 6.13. Qualitatively, this mismatch presented as agents getting stuck against walls (or each other), agents not kicking the ball after gripping it, or agents moving towards the ball without moving within range to grip the ball. The frequency of these mistakes was high enough that at least one incorrect behavior could be noticed in a 10 minute game, but infrequently enough that clear "soccer like" behaviors could be observed.

Next, DAGGER was used to augment the original training dataset as described in the previous section. Qualitatively, the learned behavior after 50 iterations of DAGGER would produce behavior that was largely indistinguishable from the synthetic behavior for many games in a row. Some mistakes, such as getting stuck in collisions, were completely eliminated, while the other mistakes would occur only rarely, and usually with less severity. This confirms that DAGGER qualitatively improves the performance of the learned model, and validates the expectation for the first experiment.

Figure 6.4 shows how increasing iterations of DAGGER affected Behavioral divergence. Since DAgger is increasing the number of training samples, a number of models trained with increasing amounts of data generated by just running more simulations of the synthetic behavior was used as a comparison. As the figure shows, the Behavioral Divergence trends downwards as the number of DAGGER iterations increase, while there is no obvious trend as the number of plain training samples is increased. This indicates that the training points generated by DAGGER are affecting Behavioral Divergence more than would be expected by increasing the training set size by an identical number of training points drawn from tracks of the synthetic behavior. Alternatively, Behavioral Divergence is sensitive to the changes the additional training points are having in the learned behavior, and not sensitive to the change (if one exists) in the learned model caused by increasing the training set size with more samples from tracks of the synthetic behavior. This confirms that Behavioral

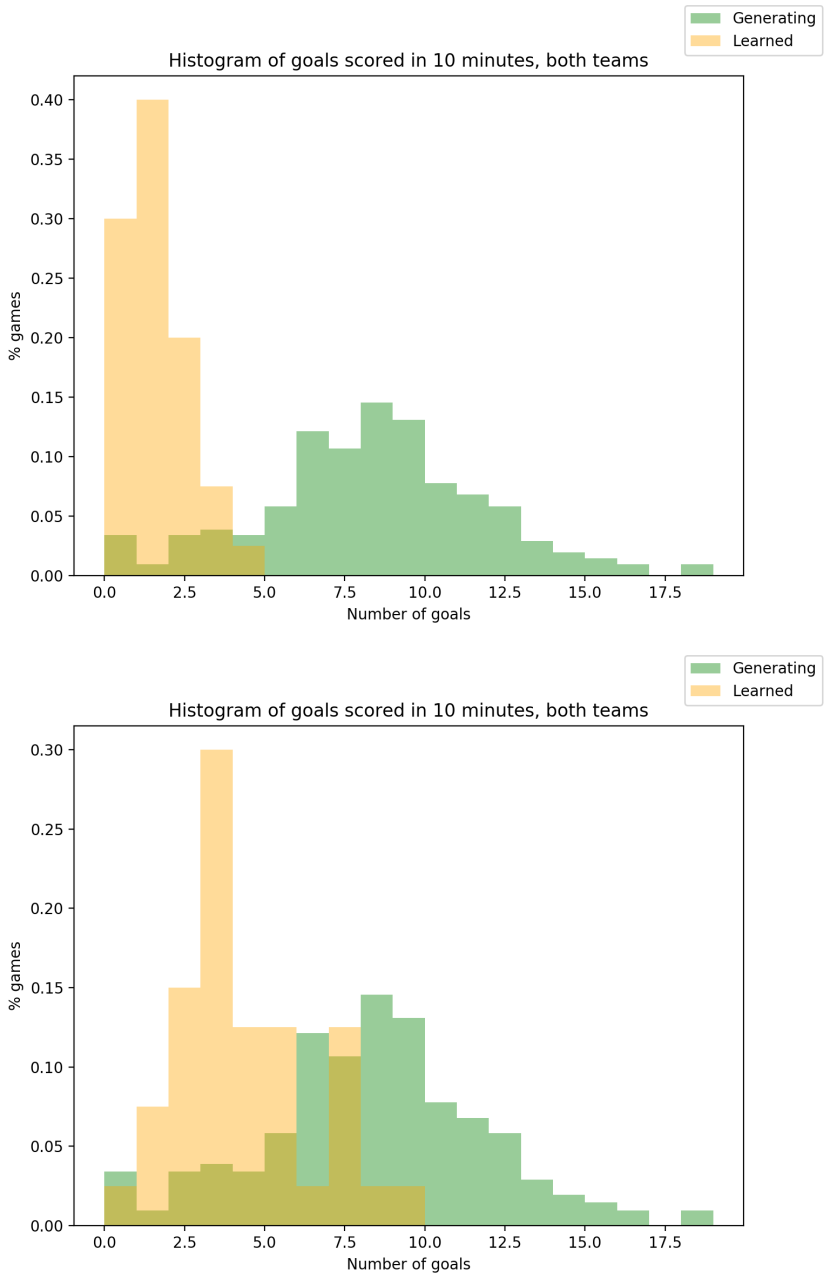

Figure 6.3: Histogram of goals scored in the first 10 minutes for target (green) and learned (orange) behavior. The top graph shows the behavior learned without DAGGER, the bottom after 200 DAGGER iterations. While not yet at parity, after 200 iterations of DAGGER the learned team is able to score more goals per 10 minute game than the initial learned team.

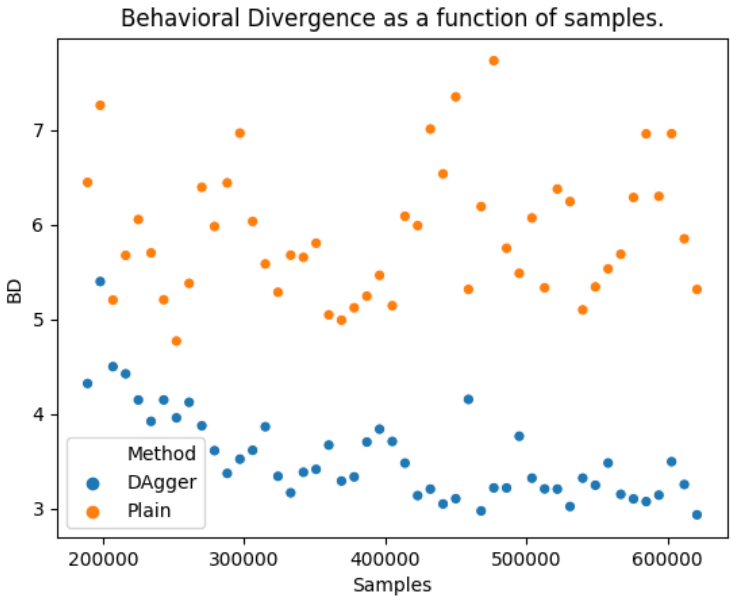

Figure 6.4: A comparison of Behavioral Divergence as the number of training samples is increased, with and without DAGGER. Increasing iterations of DAGGER correspond with a decreasing trend in Behavioral Divergence indicating improved similarity, whereas adding an equal number of training samples from additional tracks of the original data does not show any clear trend. This implies that the additional points added by DAGGER are affecting the Behavioral Divergence of the learned model more than would be expected by just increasing the amount of training data.

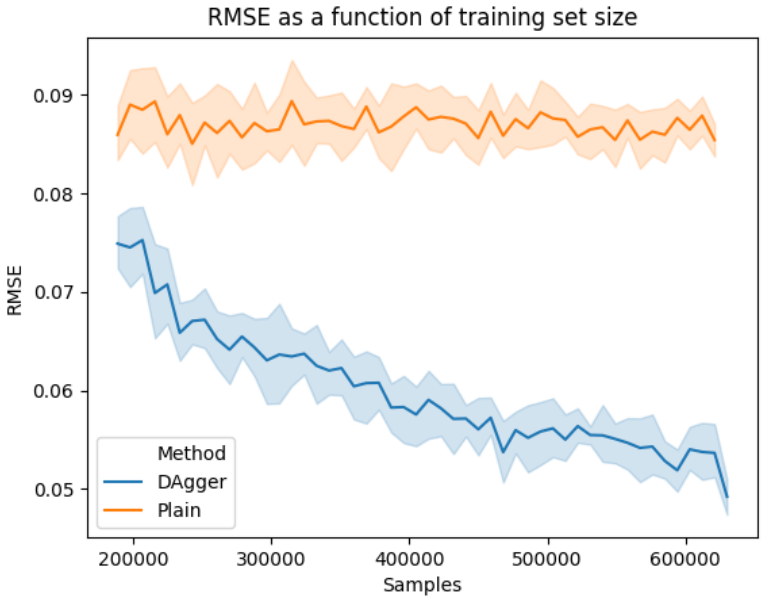

Figure 6.5: A comparison of RMSE as the number of training samples is increased, with and without DAGGER. Confidence intervals shown at 95% are computed using 10-fold cross validation. For DAGGER, there is a clear decreasing trend as the number of samples is increased, indicating that performance is improving as the number of samples increases. Without DAGGER, there is no such clear trend, which shows that the average performance *per sample* is not changing. It is important to note that for DAGGER the CV folds are across the data which it generates, which does not indicate how performance on a *withheld* test set will change as samples are increased. That is shown in Figure 6.6.

Divergence is tracking the improved qualitative performance, and validates the expectation for the second experiment.

Figure 6.5 performs the same comparison, except using RMSE in place of Behavioral Divergence. Again, performance improves as the number of DAGGER iterations increases, while remaining relatively constant with an equivalent increase in the training data using samples from tracks of the synthetic behavior, confirming the expectation of the third experiment. However the reason behind the downward trend is not the same. In the case of Behavioral Divergence, the tracks that were used to compute the histograms were independently generated by running the learned model in simulation, while for RMSE, the error was computed using cross-validation on the additional data. This means that, in the case

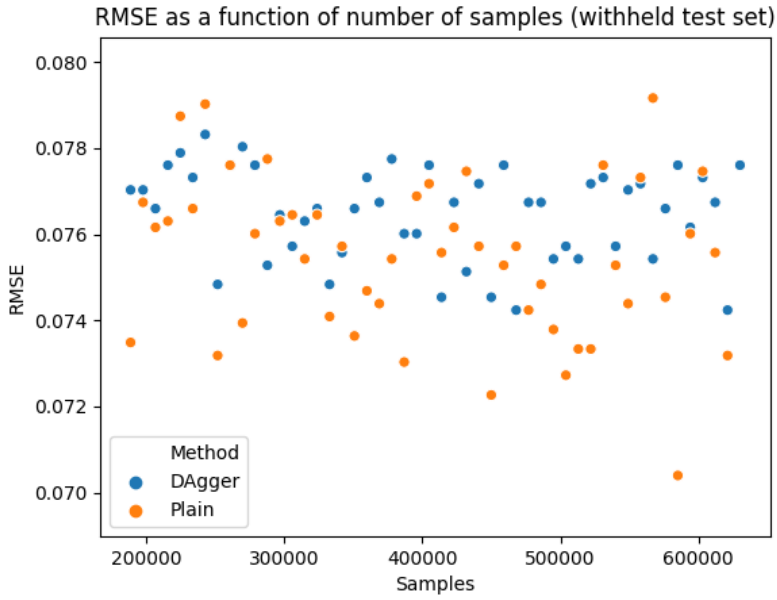

Figure 6.6: A comparison of RMSE as the number of training samples is increased on a *withheld* testing set, with and without DAGGER. Unlike Figure 6.5, there is no clear trend in performance as the number of samples is increased. This indicates that the samples that DAGGER generates states which have features that are *not* typically seen in the tracks of the "expert" or target behavior. Since the behavior learned using DAGGER is better both in terms of qualitative assessment (fewer examples of obviously incorrect behavior such as colliding with walls or missing the ball) and quantitative measures (increased average goals scored per 10 minute game), and Behavioral Divergence shows improvement on the withheld test set while RMSE does not, Behavioral Divergence is sensitive to an aspect of performance which correlates with our expectation of behavioral similarity that RMSE is not sensitive to.

of DAGGER, the cross validation was being performed across the samples being *generated* by DAGGER. The held-out fold is not used when training the model to calculate the RMSE, but the features that are part of the test set are determined by the learned behavior of the previous iteration. If the primary assumption of DAGGER is correct, the distribution of such query points can be *significantly* different from the query points generated by the synthetic behavior. For this reason it is important to examine the behavior of the models learned using DAGGER on a withheld test set, in addition to cross-validation. Figure 6.6 shows the RMSE performance as training samples are increased with and without DAGGER. The performance does not have an obvious trend as samples are increased, and whats more, the numerical values for the error are much closer than in the previous figures. This disparity indicates that the samples generated by DAGGER are not drawn from the same distribution as the samples generated by tracks of the synthetic behavior, which agrees with the expectations for the fourth experiment, and the theoretical justification for using DAGGER in the first place. This also confirms that Behavioral Divergence is sensitive to some aspects of behavior that correlate with the qualitative improvements mentioned previously, while RMSE is not.

Figures 6.7, 6.8, 6.9, 6.10, 6.11, 6.12, and 6.13 examine each of the behavioral measures used in this calculation of Behavioral Divergence, and suggest what aspects of behavior are being captured by each, and how the changes in distribution correspond to the qualitative changes that were observed. Some of the measures show significant improvements, while others do not change much from the initial model. This is in agreement with the expectations given for the fifth experiment.

While qualitatively the improvements in behavior become harder to distinguish after 50 DAGGER iterations, Behavioral Divergence does slowly improve for considerably longer, up to at least 200 iterations. Since the computational cost of running simulations of the learned models increases as the number of data points increase, and the improvements in performance were diminishing, the full comparisons with RMSE were not performed for

models past the 50th iteration of DAGGER; though the results were used to generate the histograms in the remaining figures.

Summary of comparisons between Behavioral Divergence and RMSE

This chapter explored how the Behavioral Divergence of a learned model can be expected to change as the model becomes more similar to the target behavior. That is, Behavioral Divergence (with appropriately chosen behavioral measures) decreases as similarity with the target increases. Additionally, Behavioral Divergence is complementary to RMSE, in that a model which is more similar to the target behavior may not show improved RMSE, but can show improved Behavioral Divergence. Chapter 7 concludes this dissertation with a summary of the results presented, and discusses remaining open problems and possible avenues for future work.

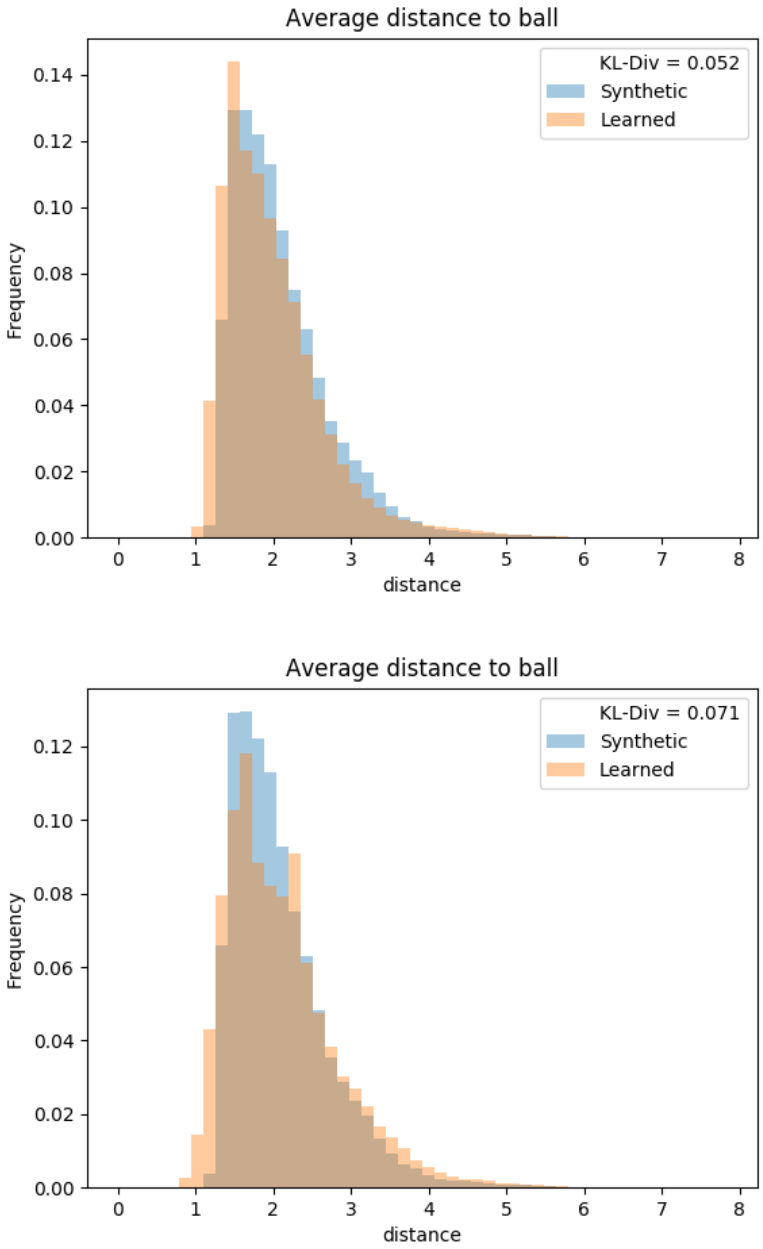

Figure 6.7: Histogram of the average distance to the ball behavioral measure, with the initial model (*top*) and after 200 dagger iterations (*bottom*). In this example, the distributions are nearly the same right from the beginning, and after running DAGGER. This indicates that the aspects of the behavior that correlate with distance to the ball are quickly learned, even without access to the additional samples generated by DAGGER.

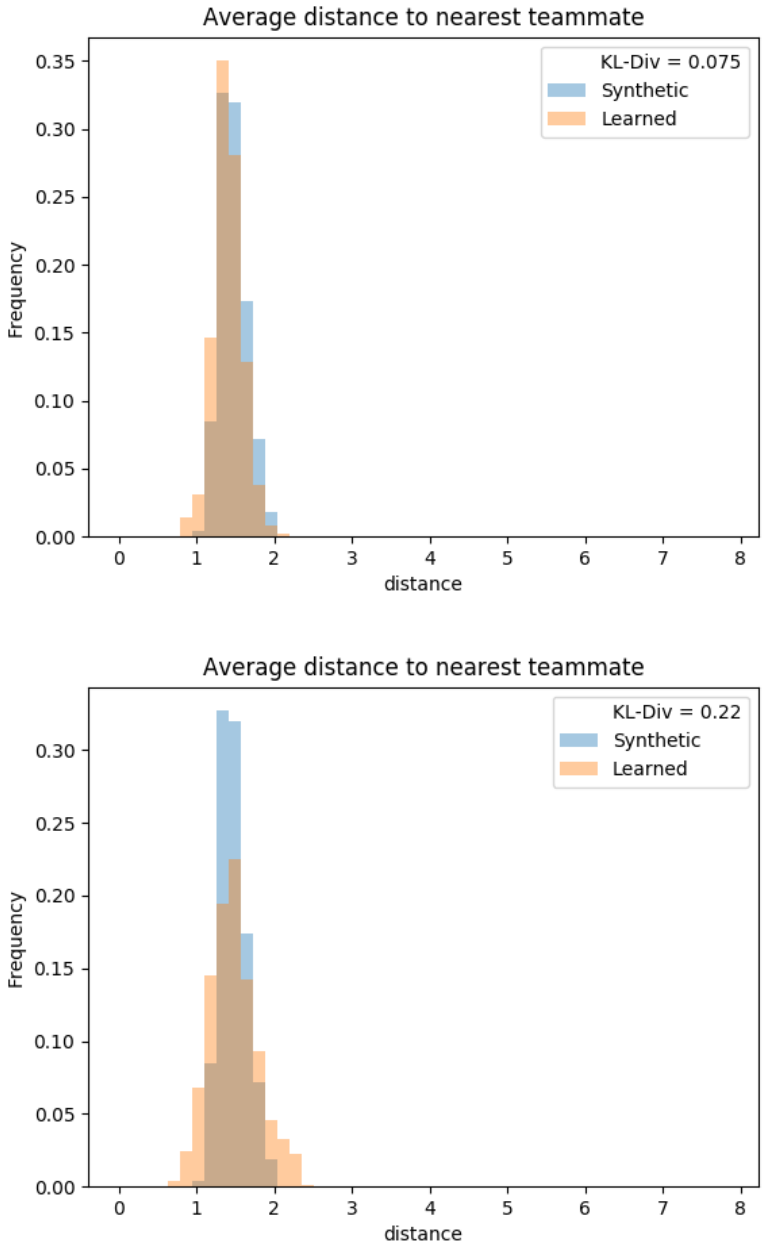

Figure 6.8: Histogram of the average distance to nearest teammate behavioral measure, with the initial model (*top*) and after 200 dagger iterations (*bottom*). In this case the learned behavior starts with a behavior that closely matches this behavioral measure, and after 200 iterations of DAGGER, the behavior is actually less similar. Since the mismatch seems to lie in mainly the tails of the learned behavior's distribution, one explanation is that the training examples being added by DAGGER could be improving performance on some parts of the feature space that are not frequently seen at the expense of performance on more common areas of the feature space.

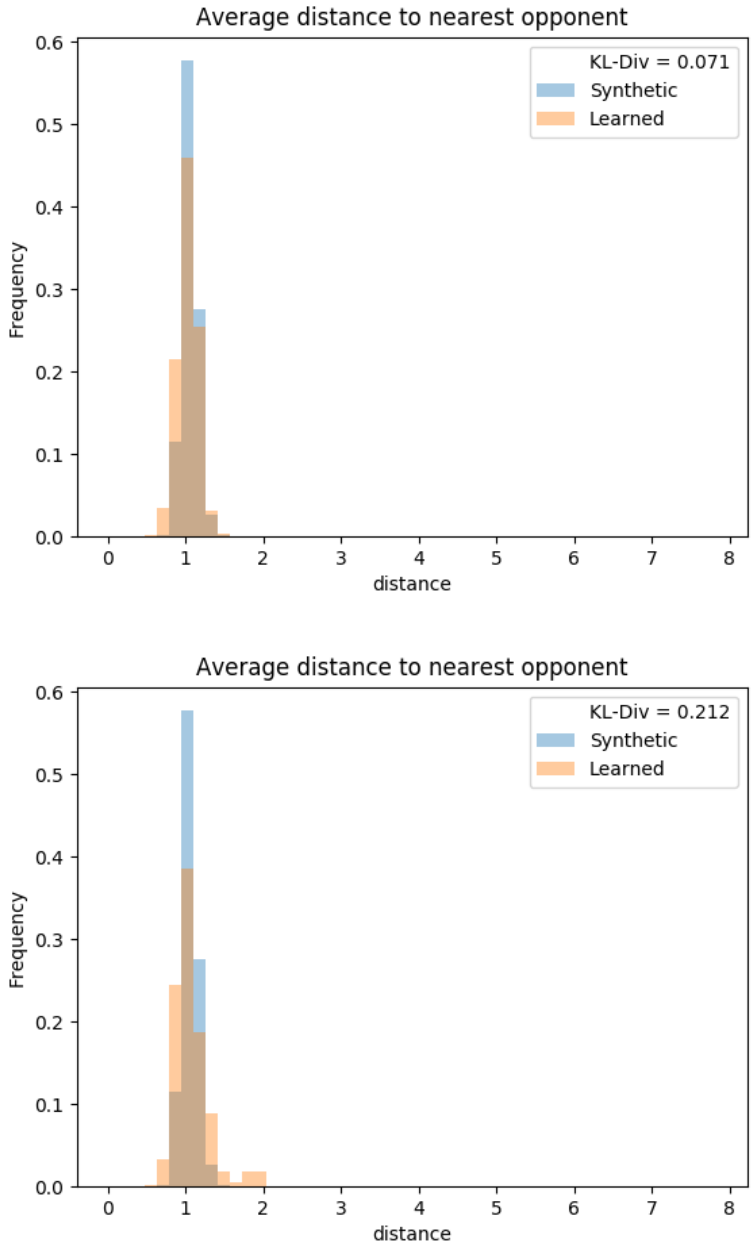

Figure 6.9: Histogram of the average distance to nearest opponent behavioral measure, with the initial model (*top*) and after 200 dagger iterations (*bottom*). This is an example where increased dagger iterations do not appear to improve the KL-divergence of this measure. Initially, the distributions are quite similar, but with increased iterations they have drifted apart. The difference is still small, relative to the improvements in the other behavioral measures, but this might indicate an aspect of the behavior that is not improving with DAGGER iterations. The dissimilarity seems to be mostly in the right tail of the learned behavior, indicating that the learned agents could be staying further away from the opposing team than the synthetic behavior does.

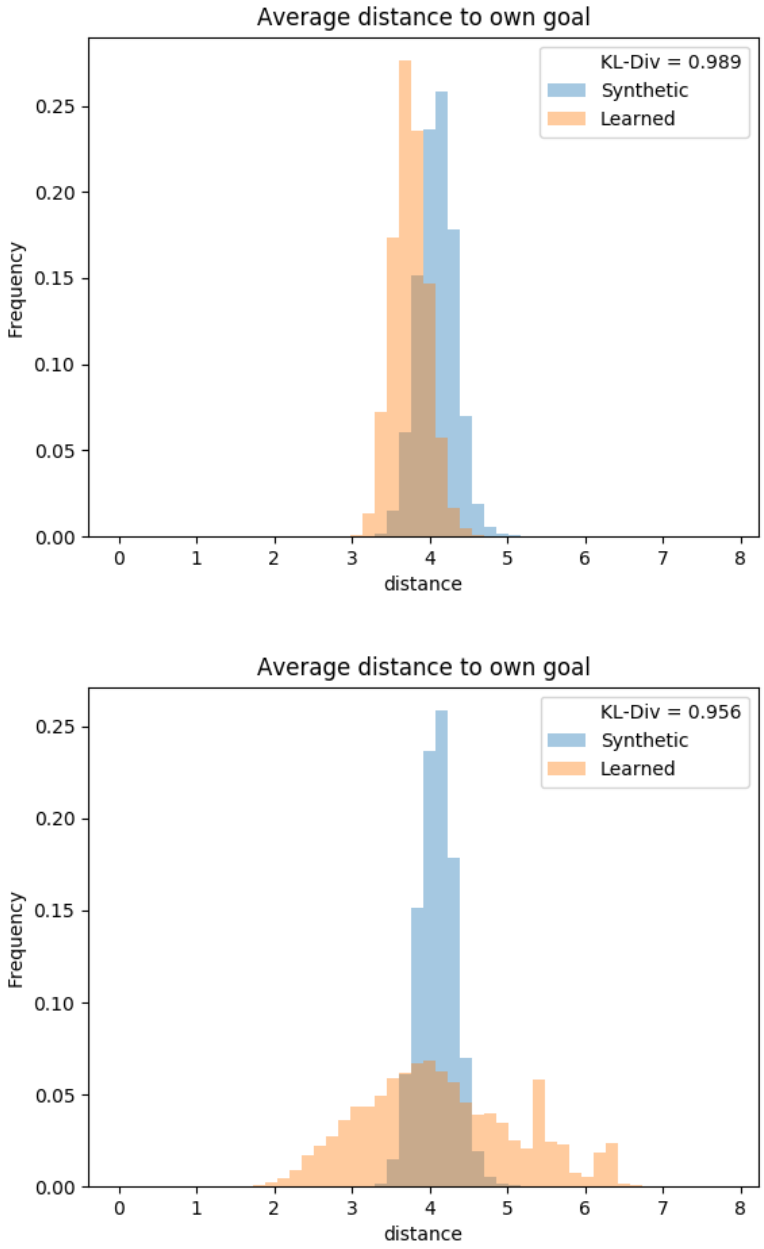

Figure 6.10: Histogram of the average distance to own goal behavioral measure, with the initial model (*top*) and after 200 dagger iterations (*bottom*). In the synthetic behavior, the majority of the agents are RECEIVERs, and tend to stay spaced out around the ball, which is usually near the center of the match, resulting in the peak near 4m. Initially, the learned behavior does not stay centered around the ball, and so drifts closer to the goal, especially when the ball is more frequently in the defensive side of the field because their opponents are actively kicking towards their goal. By the 200th iteration, the distribution is shifting to the right, although it has not reached parity.

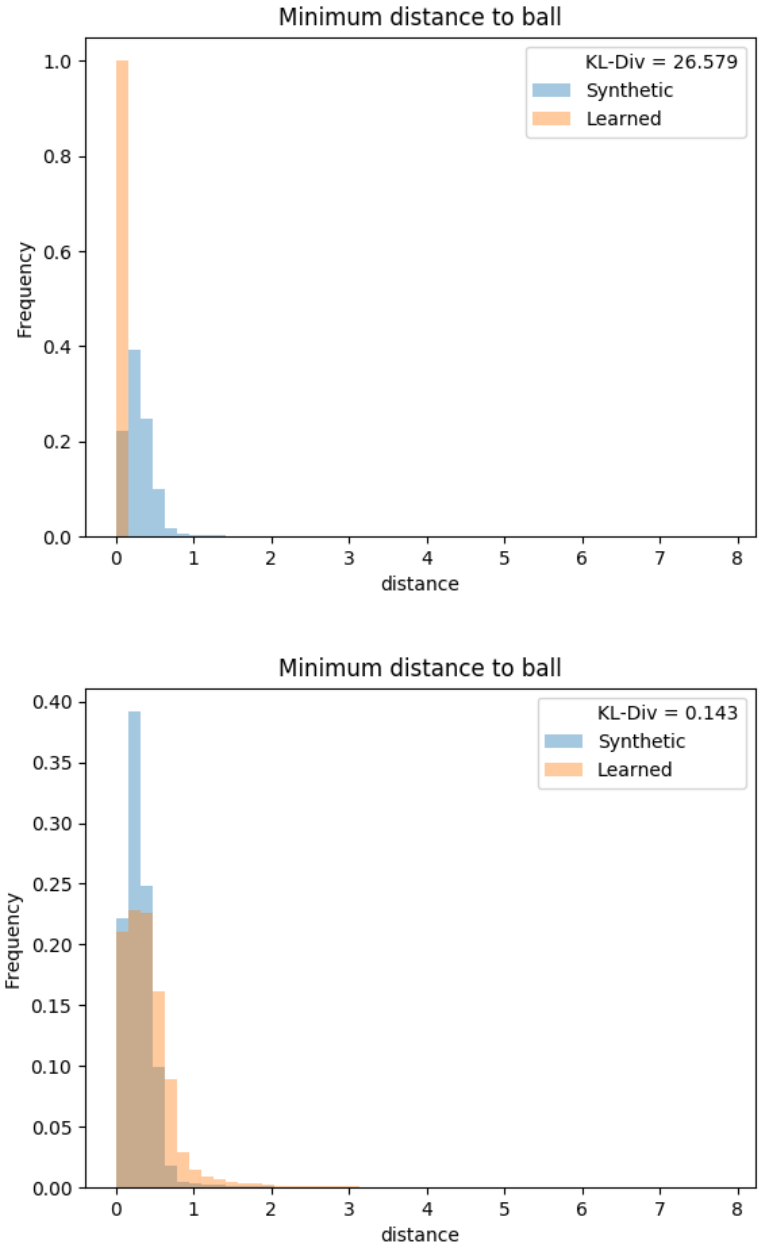

Figure 6.11: Histogram of the minimum distance to the ball behavioral measure, with the initial model (*top*) and after 200 dagger iterations (*bottom*). In the synthetic behavior, the STRIKER is the closest agent to the ball, and while it does approach the ball to within contact, it very quickly orients to a target and kicks, resulting in the distribution seen. Initially, the learned behavior will grip the ball, but will not kick it very quickly, resulting in the strong spike near zero. By the 200th iteration, the learned agents are able to quickly kick the ball.

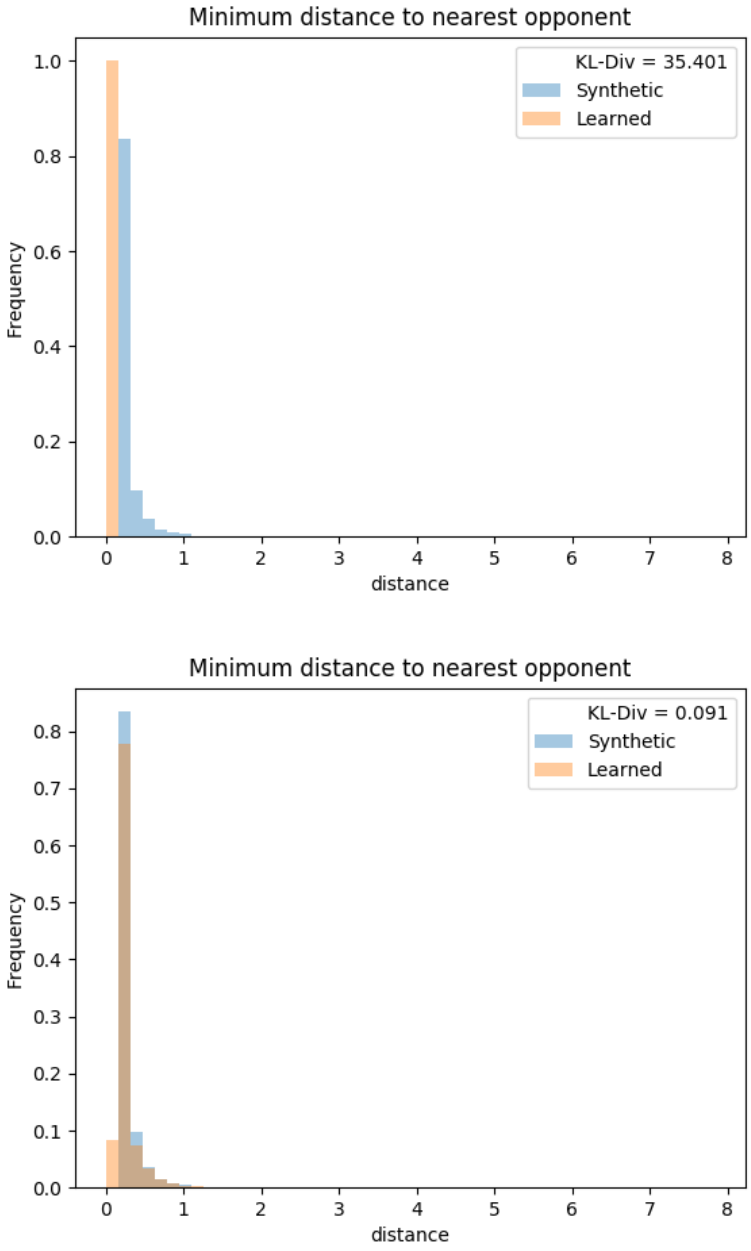

Figure 6.12: Histogram of the minimum distance to nearest opponent behavioral measure, with the initial model (*top*) and after 200 dagger iterations (*bottom*). While most of the team members are receivers in the synthetic behavior, the STRIKER does not avoid other agents, and so the minimum distance to the nearest opponent will remain relatively small for most of the time. In the initial learned behavior, the receivers do not avoid other agents very well, and so the minimum distance is usually in contact with another agent. By the 200th iteration, the learned agents are more successful at avoiding each other.

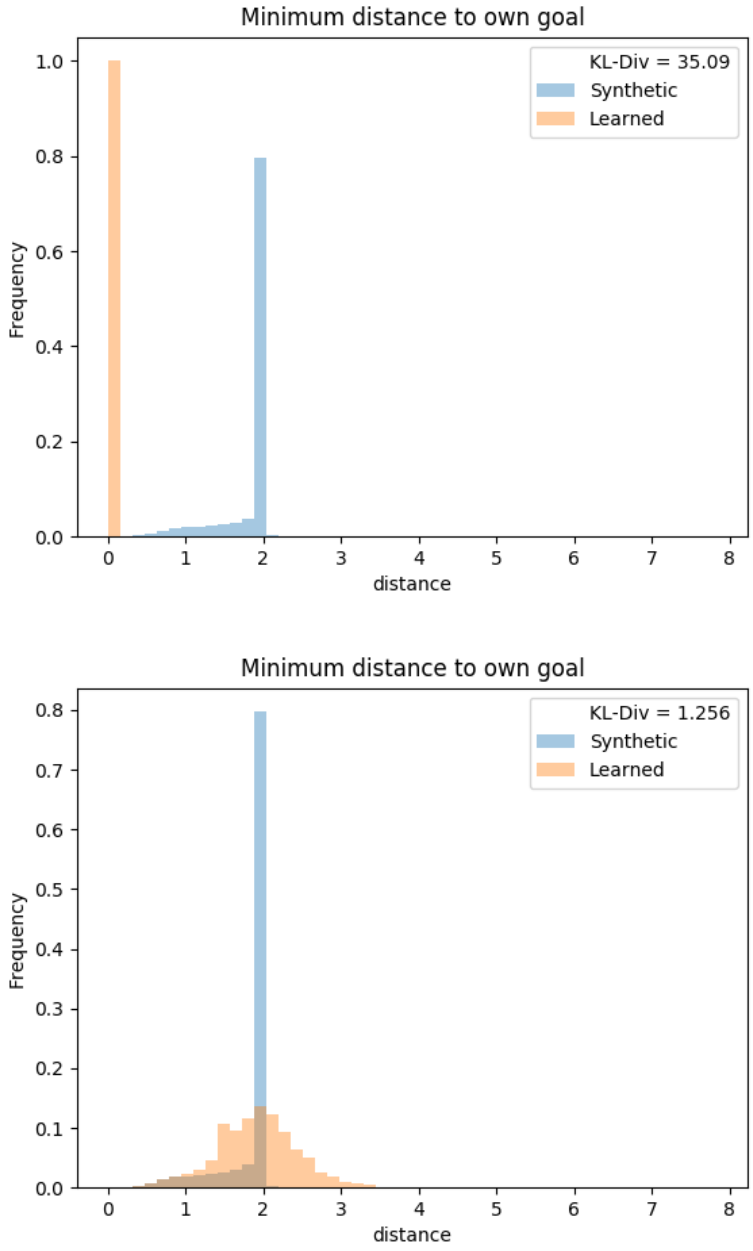

Figure 6.13: Histogram of the minimum distance to own goal behavioral measure, with the initial model (*top*) and after 200 dagger iterations (*bottom*). The synthetic behavior has at least one agent within 2 meters of the goal at all times, hence the strong spike at 2m, with lower frequency at closer ranges. The single peak in the initial learned behavior is likely due to an agent becoming stuck against the wall near the goal. By the 200th iteration, the learned behavior keeps the nearest team member around 2m from the goal, with some small error.

CHAPTER 7

CONCLUSION

The task of learning executable models of collective behavior from tracks presents a number of interesting and difficult challenges, of which this dissertation addresses a few important examples.

Contributions

1. Behavioral Divergence, a method for comparing the similarity of behaviors from their tracks which is the central contribution of this dissertation presented in chapter 5, provides a quantitative way of comparing the quality of learned models. The key insight that informs Behavioral Divergence is that the *distribution of behavior* is important in assessing similarity between two example behaviors. This insight leads naturally to a framework for learning executable models from tracks.

2. The methods presented in chapter 3 illustrate how to apply this framework to learn simple models of stochastic behaviors that do not require internal state from examples, by efficiently sampling from an estimate of the observed distribution.

3. More complex behaviors can be learned using the methods described in chapter 4, which learn models that can switch between multiple low-level behaviors.

4. Simulations run using executable models of behavior can be a valuable resource for developing and validating data analysis methods for studying aspects of the modeled agents, as was shown in chapter 2 which illustrated how a manually constructed executable model was used to help design and validate a technique for inferring the social structures of a group of monkeys from tracking data.

5. Qualitative observation supported by experiments validate that Behavioral Divergence captures a notion of similarity that is complementary to measures of predictive performance, as illustrated in chapter 6.

These contributions address an important subset of the challenges in learning these executable models, but a number of open problems remain. The primary open problem remains learning these models when tracking data contains significant noise. For many of the domains presented in this dissertation, the accuracy of models learned from real-world tracking data is significantly impacted by noise, and the noise itself is exacerbated by the compounding effect of having multiple agents. Any error in an agent's individual actions can also affect any agent reacting to those actions, and all of these errors can compound over time. When noise is low, as is the case in the simulation results, the learned models are usually able to accurately recreate the behavior in question. So in instances where it is possible to filter or pre-process the tracking data it may be possible to minimize this impact, but this is not the case for every real world scenario. A more holistic approach might be to integrate the process of learning a model of behavior into the tracking problem itself. In a sense "closing the loop" by having a multi-target tracking algorithm which improves its ability to track a group of agents by building a model of how the behave. The ability to point a fully automated device at a group of agents interacting in the world, and produce tracks of those agents as well as a fully formed executable model is compelling, and not yet realized.

Another open problem is the creation of more sophisticated behavior representations and learning methods that leverage the same insight into the importance of the distribution of behavior. One interesting avenue of research is to explore learning methods that directly optimize Behavioral Divergence as an objective. Two non-trivial obstacles to this approach are the non-convex nature of the objective, and the reliance on simulation runs to collect good estimates of the underlying behavioral measures, both of which present significant computational bottlenecks. The connection between Behavioral Divergence and the cross

entropy objective 5.5 suggests that some methods are already optimizing similar objective functions, albeit with differing assumptions. Recent work from the fields of Reinforcement Learning and Imitation Learning is beginning to examine the suitability of methods which learn distributions of behavior specifically for non-deterministic policies, likely due to the increased interest generated by the success of related work in Deep Learning in general, and Generative Adversarial Networks in particular (Goodfellow et al. 2014). It is likely that deep networks designed along the same principles as GANs would fit well into the framework presented in this dissertation.

Finally, this work does not address how the behavioral measures which form the core of Behavioral Divergence can be selected for arbitrary problem domains. In the examples given here, a set of behavioral measures is given as describing important or characteristic aspects of the behavior in question. In a sense, this answers the question of how to determine if two behaviors are similar by relying on some set of features that are determined *a priori*. While Behavioral Divergence does answer the question of how to systematically measure and combine these features, it does not address where they come from in the first place. A systematic study exploring the space of possible behavioral measures for wide categories of behavior, potentially including methods for generating or otherwise sampling from this space and particularly in a way which provides guarantees for the resulting Behavioral Divergence would be a natural extension to the presented work.

While there are a variety of open problems still to be addressed, the results presented in this dissertation demonstrate how executable models can be learned, represented, evaluated, and used.

REFERENCES

Arjovsky, Martin, Soumith Chintala, and Léon Bottou (2017). "Wasserstein Generative Adversarial Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, pp. 214–223.

Arkin, Ronald C (1998). *Behavior-based robotics*. MIT Press.

Balch, Tucker (2000). "Hierarchic social entropy: An information theoretic measure of robot group diversity". In: *Autonomous robots* 8.3, pp. 209–238.

Baum, Leonard E et al. (1970). "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains". In: *The annals of mathematical statistics* 41.1, pp. 164–171.

Bengio, Yoshua and Paolo Frasconi (1995). "An input output HMM architecture". In: *Advances in neural information processing systems*, pp. 427–434.

Cha, Sung-Hyuk (2007). "Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions". In: *International Journal of Mathematical models and Methods in Applied Sciences* 1.4, pp. 300–307.

Couzin, Iain D. et al. (2002). "Collective memory and spatial sorting in animal groups". In: *Journal of theoretical biology* 218.1, pp. 1–11.

De Gooijer, Jan G and Dawit Zerom (2003). "On conditional density estimation". In: *Statistica Neerlandica* 57.2, pp. 159–176.

Feldman, Adam, Maria Hybinette, and Tucker Balch (2012). "The multi-iterative closest point tracker: An online algorithm for tracking multiple interacting targets". In: *Journal of Field Robotics* 29.2, pp. 258–276.

Goodfellow, Ian et al. (2014). "Generative adversarial nets". In: *Advances in neural information processing systems*, pp. 2672–2680.

Gordon, Deborah M, Richard E Paul, and Karen Thorpe (1993). "What is the function of encounter patterns in ant colonies?" In: *Animal Behaviour* 45.6, pp. 1083–1100.

Hemelrijk, Charlotte K. (2000). "Towards the integration of social dominance and spatial structure". In: *Animal Behaviour* 59.5, pp. 1035–1048.

Holmes, Michael, Alex Gray, and Charles L. Isbell (2007). "Fast Nonparametric Conditional Density Estimation". In: *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence (UAI)*. (UAI).

Holmes, Michael, Alexander Gray, and Charles L. Isbell (2010). "Fast Kernel Conditional Density Estimation: A Dual Tree Monte Carlo Approach". In: *Computational Statistics and Data Analysis* 54.7, pp. 1707–1718.

Hrolenok, Brian and Tucker Balch (2013a). "Learning Executable Models of Multiagent Behavior from Live Animal Observation". In: *ICML 2013 Workshop on Machine Learning For System Identification*.

— (2013b). "Learning Schooling Behavior from Observation". In: *Advances in Artificial Life, ECAL*. Vol. 12, pp. 686–691.

— (2014). "Assessing learned models of fish schooling behavior". In: *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, pp. 1435–1436.

Hrolenok, Brian, Byron Boots, and Tucker Hybinette Balch (2017). *Sampling Beats Fixed Estimate Predictors for Cloning Stochastic Behavior in Multiagent Systems*.

Hrolenok, Brian, Hanuma Teja Maddali, et al. (2014). "Inferring Social Structure of Animal Groups From Tracking Data". In: *ALIFE 14: The Fourteenth Conference on the Synthesis and Simulation of Living Systems*. Vol. 14, pp. 336–343.

Huang, Pipei et al. (2012). "Learning a projective mapping to locate animals in video using RFID". In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, pp. 3830–3836.

Hyndman, Rob J, David M Bashtannyk, and Gary K Grunwald (1996). "Estimating and visualizing conditional densities". In: *Journal of Computational and Graphical Statistics* 5.4, pp. 315–336.

Katz, Yael et al. (2011). "Inferring the structure and dynamics of interactions in schooling fish". In: *Proceedings of the National Academy of Sciences* 108.46, pp. 18720–18725.

Kullback, Solomon and Richard A Leibler (1951). "On information and sufficiency". In: *The annals of mathematical statistics* 22.1, pp. 79–86.

List, Christian, Christian Elsholtz, and Thomas D Seeley (2009). "Independence and interdependence in collective decision making: an agent-based model of nest-site choice by honeybee swarms". In: *Philosophical transactions of The Royal Society B: biological sciences* 364.1518, pp. 755–762.

Liu, Han, John Lafferty, and Larry Wasserman (2007). "Sparse nonparametric density estimation in high dimensions using the rodeo". In: *Artificial Intelligence and Statistics*, pp. 283–290.

Martin, Paul and Paul Patrick Gordon Bateson (1993). *Measuring behaviour: an introductory guide*. Cambridge University Press.

Muja, Marius and David G Lowe (2014). "Scalable nearest neighbor algorithms for high dimensional data". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 11, pp. 2227–2240.

Pratt, Stephen C et al. (2005). "An agent-based model of collective nest choice by the ant *Temnothorax albipennis*". In: *Animal Behaviour* 70.5, pp. 1023–1036.

Rabiner, Lawrence R (1989). "A tutorial on hidden Markov models and selected applications in speech recognition". In: *Proceedings of the IEEE* 77.2, pp. 257–286.

Reynolds, Craig W (1987). "Flocks, herds and schools: A distributed behavioral model". In: *ACM SIGGRAPH Computer Graphics*. Vol. 21, pp. 25–34.

Ross, Stéphane and Drew Bagnell (2010). "Efficient reductions for imitation learning". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 661–668.

Ross, Stéphane, Geoffrey Gordon, and Drew Bagnell (2011). "A reduction of imitation learning and structured prediction to no-regret online learning". In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635.

Rubner, Yossi, Carlo Tomasi, and Leonidas J Guibas (2000). "The earth mover's distance as a metric for image retrieval". In: *International journal of computer vision* 40.2, pp. 99–121.

Sapolsky, Robert M. (2005). "The Influence of Social Hierarchy on Primate Health". In: *Science* 308.5722, pp. 648–652. eprint: `http://www.sciencemag.org/content/308/5722/648.full.pdf`.

Sbalzarini, Ivo F. and Petros Koumoutsakos (2005). "Feature point tracking and trajectory analysis for video imaging in cell biology". In: *Journal of structural biology* 151.2, pp. 182–195.

Schleidt, Wolfgang M et al. (1984). "A proposal for a standard ethogram, exemplified by an ethogram of the bluebreasted quail (coturnix chinensis) 1". In: *Zeitschrift für Tierpsychologie* 64.3, pp. 193–220.

Stephens, Shannon B.Z. and Kim Wallen (2013). "Environmental and social influences on neuroendocrine puberty and behavior in macaques and other nonhuman primates". In: *Hormones and Behavior* 64.2. Puberty and Adolescence, pp. 226–239.

Tamar, Aviv et al. (2016). "Value iteration networks". In: *Advances in Neural Information Processing Systems*, pp. 2154–2162.

Venkatraman, Arun, Martial Hebert, and J Andrew Bagnell (2015). "Improving multi-step prediction of learned time series models". In: *AAAI (Awaiting Publication)*.

Wallen, Kim (1996). "Nature Needs Nurture: The Interaction of Hormonal and Social Influences on the Development of Behavioral Sex Differences in Rhesus Monkeys". In: *Hormones and Behavior* 30.4, pp. 364–378.

Yang, Yu-Ting et al. (2012). "Ant Hunt: Towards a Validated Model of Live Ant Hunting Behavior." In: *FLAIRS Conference*.

Zhan, Eric et al. (2018). "Generative Multi-Agent Behavioral Cloning". In: *CoRR* abs/1803.07612. arXiv: 1803.07612.

Zhao, Junbo, Michael Mathieu, and Yann LeCun (2016). "Energy-based generative adversarial network". In: *arXiv preprint arXiv:1609.03126*.